Reconstruction of Sculptured Surface Using Coordinate Measuring Machines[1]

Yuan C. Hsieh

UUCS-93-010

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

April 22, 1993

Abstract

This paper presents a strategy for reverse engineering that uses a coordinate measuring machine to reconstruct three-dimensional sculptured surfaces. A rough initial model of the surface is generated manually. An iterative method is then used to refine the surface model until the error is within a desired bound. The reverse engineering process is broken down into three phases: data acquisition, surface reconstruction and surface evaluation. For data acquisition, an exhaustive search algorithm is used to find a safe probe orientation in the vicinity of the target surface, and a coarse cell decomposition method is followed to manipulate the coordinate measuring machine in its work space. Surfaces are modeled using a B-spline approximation technique. The position difference between the surface model and the measured data is used as a simple criterion to evaluate the quality of the reconstructed surface model.

Several examples of the use of this technique are presented, including a sculptured pocket, a model of compressor blade surfaces, and two physical models of the human bones. Criteria for evaluating the performance of the obstacle avoidance algorithm are discussed and the results are presented. In addition, the quality of the surface models is also presented.

[1]This work was supported in part by DARPA (N00014-91-J-4123). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

RECONSTRUCTION OF SCULPTURED SURFACES USING COORDINATE MEASURING MACHINES

by

Yuan C. Hsieh

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

The University of Utah

June 1993

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Yuan C. Hsieh

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Samuel H. Drake

Sanford G. Meek

Thomas C. Henderson

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of The University of Utah:

I have read the thesis of Yuan C. Hsieh in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to the Graduate School.

Date

Samuel H. Drake
Chair, Supervisory Committee

Approved for the Major Department

Robert B. Roemer
Chair/Dean

Approved for the Graduate Council

B. Gale Dick
Dean of The Graduate School

ABSTRACT

This paper presents a strategy for reverse engineering that uses a coordinate measuring machine to reconstruct three-dimensional sculptured surfaces. A rough initial model of the surface is generated manually. An iterative method is then used to refine the surface model until the error is within a desired bound. The reverse engineering process is broken down into three phases: data acquisition, surface reconstruction and surface evaluation. For data acquisition, an exhaustive search algorithm is used to find a safe probe orientation in the vicinity of the target surface, and a coarse cell decomposition method is followed to manipulate the coordinate measuring machine in its work space. Surfaces are modeled using a B-spline approximation technique. The position difference between the surface model and the measured data is used as a simple criterion to evaluate the quality of the reconstructed surface model.

Several examples of the use of this technique are presented, including a sculptured pocket, a model of compressor blade surfaces, and two physical models of the human bones. Criteria for evaluating the performance of the obstacle avoidance algorithm are discussed and the results are presented. In addition, the quality of the surface models is also presented.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

The author wishes to thank the member of the Alpha_1 project, especially Dr. Richard Riesenfeld, Dr. Elaine Cohen and Dr. Elizabeth Cobb for their support and inputs throughout the research. In addition, the author also likes to thank the committee members for their suggestions on the manuscript. Finally, the author wants to acknowledge the following people for their help at various stages of the research and preparing the thesis, Dr. Shawn Cunninghan and Cynthia Eaton.

This work was supported in part by DARPA (N00014-91-J-4123). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

CHAPTER 1

INTRODUCTION

Reverse engineering is the task of accurately producing a computational and functional model from an existing physical object. Because it is usually the case that one wants to make the measured object, accuracy of the model is paramount. This research focuses on a subset of the reverse engineering task, that of using a coordinate measuring machine (CMM) to measure sculptured surfaces and to reconstruct these surfaces into CAD/CAM models. There are two stages for a typical reverse engineering system: data acquisition and object modeling. The work presented here combines those stages into a single integrated process that iterately updates the model based on acquired data, and new samples are estimated based on the current model.

Data acquisition methods can be classified into two categories: remote sensing and tactile sensing. Remote sensors include cameras and range finders. Cameras usually produce digitized gray-scale intensity images as sensor data, and range finders produce depth maps, which are arrays of values that describe the distances from the sensor to the object within the field of view. Remote sensing methods can acquire data in a short amount of time and produce dense data sets. Tactile sensors acquire data by physically touching the object. Coordinate measuring machines are an example of tactile sensors. Tactile sensing techniques are usually time consuming and produce sparse data; however, the accuracy of data produced using CMMs currently surpasses that of commonly available remote sensing techniques.

Object modeling requires a scheme to represent the object and methods to reconstruct the object from measured data. There are many methods to represent three-dimensional objects [1, 2]. One particular method of interest is the surface boundary representation. The surface boundary representation method defines the object by defining the three-dimensional surfaces that bound the object. Three-dimensional surfaces can be represented using explicit functions, one specific form being the parametric function:

$$S = \{(x, y, z) : x = h(u, v), y = g(u, v), z = f(u, v), (u, v) \subset \mathbf{D}, \Re^2 \supseteq \mathbf{D}\}. \tag{1.1}$$

B-spline surface representation is an example of parametric functions.

Unlike remote sensing approaches, the most difficult aspect of data acquisition using the CMM is obstacle avoidance. Because measurements using the CMM require controlled contact of the probe with the object, the CMM will come in close proximity of the object, and the risk of unexpected collisions is high. Without a robust algorithm, collision and damage to the object or the CMM are possible. Also, there is no reason to perform surface reconstruction if data cannot be acquired accurately and consistently. In addition, a safe obstacle avoidance algorithm can be used for automated inspection tasks. The quality of the model depends on the data acquired, and the reliability of the data acquisition algorithm and a plan for data sampling depend on the estimated surface model. Therefore, the two phases of the reverse engineering process are interdependent.

The reverse engineering process is developed and integrated into the Alpha_1 geometric modeling system [3]. Alpha_1 modeling system is developed by the Alpha_1 project in the University of Utah. This integration enables the reverse engineering process to benefit from the interaction, simulation, representation, and manufacturing capabilities of the modeling system. Hence, the integrated capability in Alpha_1 supports the strategy adopted in this research.

For the remainder of this thesis, *object* will always refer to the physical piece being measured. *Obstacles* are physical objects that are potential collision threats.

Example of *obstacles* are the *object* itself, the calibration ball and supporting structures used to secure the *object*. *Model* will refer to estimated computer model of the *object*, and surface normal refers to the normal vector of a point on a surface.

In Chapter 2, a surface reconstruction process(**SuRP**) using the CMM is presented, as well as a brief survey of other approaches. In Chapter 3, data acquisition issues using the CMM, including path planning and obstacle avoidance problems, are discussed and an implementation method is presented. Chapter 4 discusses the problems and solutions of surface reconstruction using sparse data sets. In Chapter 5, problems of verification and evaluation are discussed. Experimental results and performance evaluation are presented in Chapter 6. Finally, the conclusion and the potential research topics are identified in Chapters 7 and 8.

CHAPTER 2

OVERVIEW

Other attempts have been made to reconstruct three-dimensional objects using a variety of techniques as described in [4], [5], [6], [7], [8], and [9]. There are other researches using tactile sensors for object recognition and sensor fusion tasks [10, 11, 12]. However, CMMs are not currently a favored device for reverse engineering tasks. Many feel that it is too time consuming, even though the precision of the measurements surpass that of the range data. Currently, CMMs are used in a variety of inspection and quality control tasks as described in [13], [14], [15] and [16]. A CAD/CAM system capable of manufacturing an object measured by a CMM was proposed by Lee, Chen and Lin [17]. Their system employed a two-stage surface fitting technique. They use Ferguson surface with a set of coarse points and then a least square method when they have recovered enough points. Their system uses a three-axis CMM and compensates for probe error. Kwok and Eagle [18] produced a reverse engineering system using a three-axis CMM to measure features, such as lines, planes and circles. However, their work focused on linking the CMM with existing CAD software, and measurements were accomplished manually.

In section 2.1 and section 2.2, the CMM used for this research is described, and the surface reconstruction system is presented.

2.1 Coordinate Measuring Machine

CMMs are typically five-axis robots, which are capable of moving in three-dimensional cartesian space, as well as providing roll and pitch rotation of the probe at the end of the actuator. It acquires data by physical contact using a probe. This research uses a COordinate MEasuring RObot (COMERO) CMM manufactured by Fanamation in Compton, California. The COMERO is built specifically for inspection and can only be controlled using a built-in Coordinate Measurement Inspection Software(CMIS). CMIS is an implementation of the Dimensional Measuring Interface Specification (DMIS) standard (ANSI/CAM-I 101-1990) developed by Computer Aided Manufacturing International (CAM-I) for CMMs [19]. The only sensor used in this reasearch is the PH10 touch probe from Renishaw.

Components of the probe are shown and defined in Figure 2.1. The physical size of the *cylinder*, *joint* and *z-beam* component are fixed, but different sizes of the *stylus* can be utilized depending on the object being measured. The cylinder is a cylinder 3 inches long and 1 inch in diameter, and the joint is a sphere 2.37-inch in diameter. The z-beam can be approximated by a 2.37-inch by 2.37-inch by 20-inch box. A stylus is defined by its length and the diameter of the *stylus ball*.

The stylus ball is part of the stylus, and data acquisition occurs when the stylus contacts the *object*. The center of the stylus ball is considered the *contacting position* ($\vec{\textbf{conpos}}$), but not every contacting position is valid. A contacting position is valid if and only if the contact occurred between the stylus ball and the *object*.

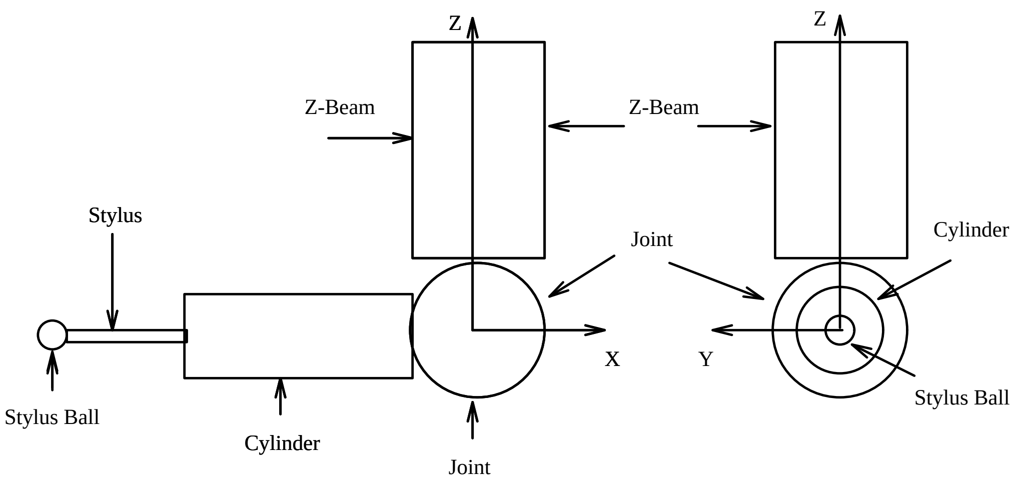

Figure 2.1. Probe components

If the contacting position is valid, then this position is the *measurement position* ($\vec{\textbf{meapos}}$). Verification of the contacting position is presented in section 5.2.1. It should be clear that the measurement position is never the same as the *real position* ($\vec{\textbf{realpos}}$) of the surface, and this difference between the measurement position and real position is the *probe error*. Figure 2.2 illustrates the differences between various positions. For a valid contact, the probe error ($\vec{\textbf{proerr}}$) is the distance between the real position and the contact position.

$$\vec{\textbf{proerr}} = \vec{\textbf{meapos}} - \vec{\textbf{realpos}}. \tag{2.1}$$

For a valid contact, the length of $\vec{\textbf{proerr}}$ is the radius of the stylus ball, and the direction is the surface normal of $\vec{\textbf{realpos}}$. Because the radius of the stylus ball is known, the surface normal of $\vec{\textbf{realpos}}$ needs to be determined to recover $\vec{\textbf{realpos}}$. One approach is to approximate the surface normal of $\vec{\textbf{realpos}}$ by measuring three points around the $\vec{\textbf{realpos}}$. The surface normal would be the normal vector of the

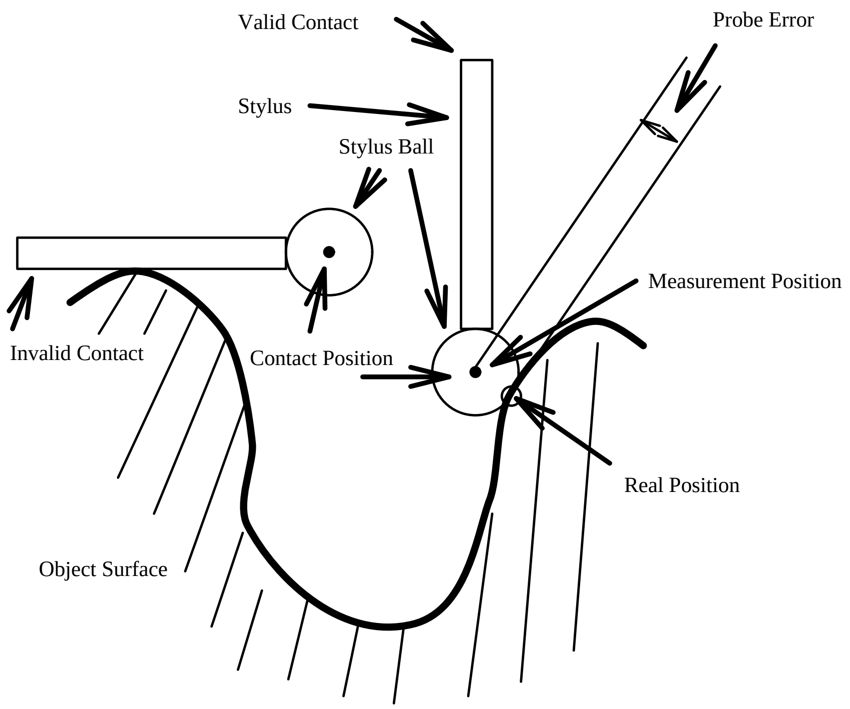

Figure 2.2. Valid and invalid probe contacting positions

plane formed by these three points. The alternative approach is to use the surface model of the *model*. With this approach, the quality of the *model* needs to be assured for the resultant $\vec{\textbf{realpos}}$ to be meaningful.

The range of motion in the COMERO coordinate system is from 0 to 40 inches in the X and Y directions, and 0 to 20 inches in the Z direction. The range of roll angles (rotation about Z axis) is 360° in 7.5° increments, and the range of pitch angles is between 0° and 105° also in increments of 7.5°. The rotational limits are shown in Figure 2.3. A measurement program written in CMIS is needed to run COMERO. CMIS provides the ability to manipulate the probe under five-axis control and to measure points or other features given an approximate position and a search space. Because CMIS was developed for inspection, there are some limitations when it is used for reverse engineering tasks. CMIS programs are top-down programs, with no built-in conditional branch and error recovery mechanisms. In other words, the program is written in the form of "Do A, Do B, Do C ...etc", If, for instance, task B cannot be accomplished for some reason, the system either crashes or stalls, and no

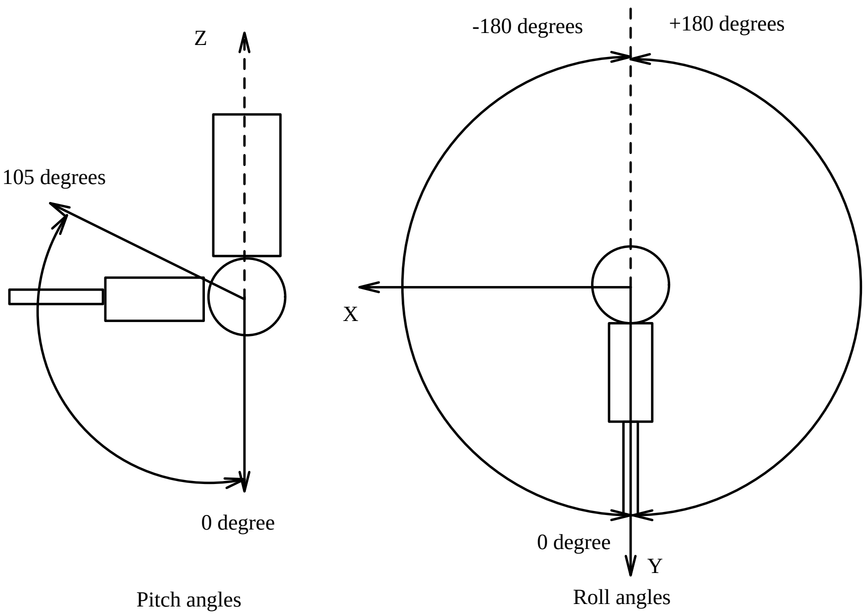

Figure 2.3. Pitch and roll rotational limit of the probe

further measurements are possible. Because CMIS is used for reverse engineering, it is possible that certain statements in the measurement program cannot be executed because of an inaccurate knowledge of the environment. All possible scenerios of invalid measurements are investigated in section 5.2.1 Another limitation of working in the CMIS environment is that measurement tasks cannot be performed in real-time.

2.2 SuRP Architecture

The proposed methodology recognizes the limitations of the CMM and uses an iterative strategy. At each iteration, the following tasks are performed: the available data are considered, regions to refine are decided upon, some new samples are selected, a CMIS program is produced to measure new samples, the CMIS program is executed, and new data are verified and combined with previous data for next iteration. The schematic of the system is presented in Figure 2.4 and a step by step description follows:

1. The first step is to acquire a rough *model* of the *object*. This initial estimate could be the result of some remote sensing system for a fully automated system. However, presently, the data are acquired from operator controlled measurements of the *object*. The initial *model* should include extremal points of the *object* to ensure the high frequency components of the *object* are included.

2. With these initial data, a *model* is produced. **SuRP** uses a B-spline surface approximation function to estimate the *model*. Function $\vec{\mathbf{Ps}}(u,v)$ defines a position on the *model*, given the parameter (u,v). A description of this procedure and the function, $\vec{\mathbf{Ps}}(u,v)$, are presented in Chapter 4.

3. The *model* is evaluated to test if the *model* passes the model tolerance criteria and the operator's subjective analysis. To evaluate the *model*, a correspon-

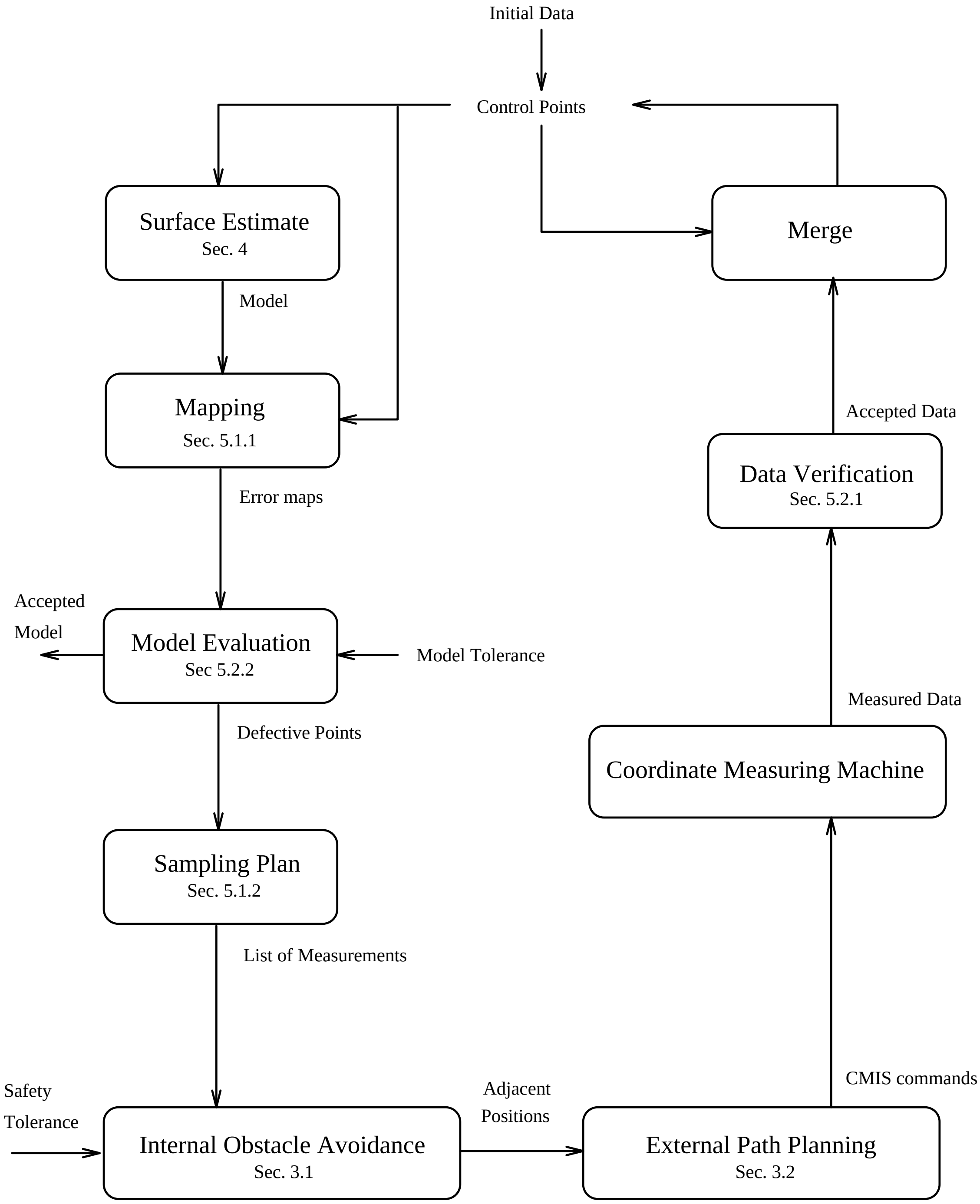

Figure 2.4. **SuRP** system architecture

dance between measured data and the *model* must be established. This mapping process is presented in section 5.1.1. Evaluation criteria and algorithms are presented in section 5.2.2.

4. If the results of the evaluation show that some regions of the model still need improvement, a set of new points on the surface in those regions is selected for measurement. This is discussed in section 5.1.2.

5. Given these sets of points, it is usually possible to plan a collision free path in order to measure these positions. This is discussed in Chapter 3. If is is not possible to find a path for some proposed measurements; these positions are not measured.

6. Once new data are acquired, it is verified as a valid measurement of the *object*. This is discussed in section 5.2.1.

7. Finally, new and old data are merged and a new iteration is initiated by returning to step 2.

CHAPTER 3

DATA ACQUISITION

The main problems of data acquisition using the CMM are obstacle avoidance and path planning, important issues in the mobile robots and automation communities. Published collision avoidance algorithms can be classified into two categories: free space methods [20, 21, 22, 23] and potential methods [24, 25, 26].

In the free space method, the work space is partitioned into free spaces and obstacles. The algorithm then searches for a path within the free spaces. Free space approaches are guaranteed to find a collision free path, if the path exists. However, computation time increases exponentially as the degrees of freedom of the robot increase. Most of the research using free space method concerns the representation of the free space, such as freeways described by Brooks [20] or cell decomposition method described by Zhu and Latombe [22].

On the other hand, the potential method typically searches for a collision free path by utilizing two potential functions. The first function, the progress function, determines the progress toward the destination, and the other function, the obstacle avoidance function, determines the distance between the manipulator and obstacles. The most difficult aspect of the potential method is the definition of the potential function for obstacle avoidance, especially for complex obstacles with free form surfaces. Another disadvantage of the potential method is that a path is not guaranteed, due to local minima that may exist in the obstacle avoidance potential functions. Most of the research involves the definition of the potential function using polyhedral obstacles. However, Kim and Khosla [26] used concepts from the theory of incompressible fluid dynamics and introduced a potential function and a panel method to approximate the potential field with complex obstacles.

To evaluate the usefulness of various published techniques in the CMM environment, it is important to understand how the CMM moves within its workspace and the specific applications. For a given probe orientation, the CMM moves from one position to another in a straight line. To change the probe orientation, the probe is rotated about the joint as shown in Figure 2.3. It is not possible to change probe orientation and move in three-dimensional Cartesian space at the same time due to the limitations of the CMIS environment. A typical CMM movement sequence to measure a point consists of the following steps as outlined in [19] and [27], and it is shown in Figure 3.1:

1. From a prior position, the probe is moved to the *approach position.* The approach position ($\vec{\mathbf{apos}}$) is defined by equation 3.1.

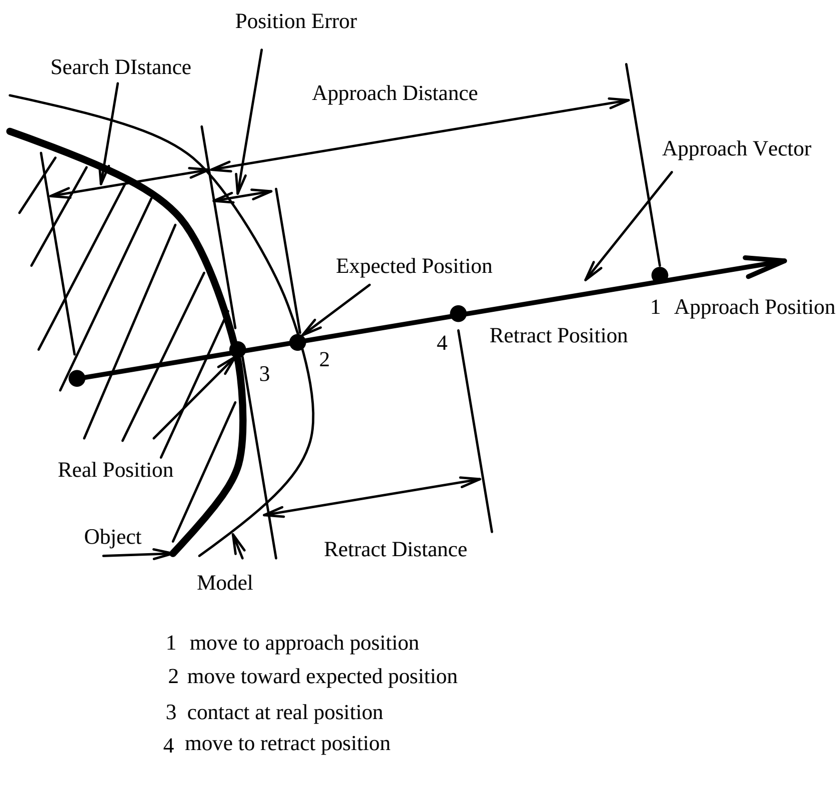

Figure 3.1. A CMIS measurement sequence

$$\vec{\mathbf{apos}} = \vec{\mathbf{epos}} + adist \times \vec{\mathbf{avec}}, \tag{3.1}$$

where $\vec{\mathbf{epos}}$ is the *expected position.* Expected position is where the measurement is expected, and it is derived from the *model.* For **SuRP**, the *expected position* is found using equation /refeq-epos.

$$\vec{\mathbf{epos}} = \vec{\mathbf{Ps}}(u, v), \tag{3.2}$$

where $\vec{\mathbf{Ps}}(u, v)$ is the equation of the surface model. $\vec{\mathbf{Avec}}$, the *approach vector*, is a unit vector pointing from $\vec{\mathbf{epos}}$ toward $\vec{\mathbf{apos}}$. The $\vec{\mathbf{avec}}$ and *adist*, the *approach distance*, are defined in the CMIS program by the user.

2. From $\vec{\mathbf{apos}}$, the probe is moved toward $\vec{\mathbf{epos}}$ in a straight line, along $\vec{\mathbf{avec}}$ in the opposite direction. The probe advances until contact with the object is detected.

   The probe is allowed to search for the object along the $\vec{\mathbf{avec}}$ within a specified *search distance* (*sdist*). *Sdist* is also defined in the CMIS program by the user. If no contact are detected at the end of the search distance, the CMIS program halts.

3. Once contact is established, the probe moves away from the object in the direction of $\vec{\mathbf{avec}}$ to the *retract position* ($\vec{\mathbf{repos}}$). The retract position is defined by equation 3.3.

$$\vec{\mathbf{repos}} = \vec{\mathbf{meapos}} + redist \times \vec{\mathbf{avec}}, \tag{3.3}$$

   where *redist* is the *retract distance* specified by the user in the CMIS program.

This research adopted an approach for data acquisition similar to the global and local planner proposed by Hwang and Ahuja [25] by breaking it down into two components: the external path planning and the internal obstacle avoidance. *Obstacles* are represented by their bounding boxes. External path planning deals with movements of the probe from one position to another outside the bounding boxes and will utilize free space method, because it guarantees a collision free path. Internal obstacle avoidance investigates the probe path within an obstacle bounding box in order to perform measurements and uses an exhaustive search method that combines a hypothesis and test algorithm and potential field concepts.

3.1 Internal Obstacle Avoidance

The internal obstacle avoidance scheme attempts to find an *approach probe* and a *measurement probe* to measure a specific position on the *object*. A measurement probe is defined by its expected position and the *probe orientation*. An approach probe is defined by the measurement probe, the approach position, and the search distance. Probe orientation is the pitch and roll angle of the probe. For simplicity, the retract distance is always set to equal the approach distance. Once these parameters are determined for a single measurement, the movement of the probe is restricted to the steps shown in Figure 3.1. Therefore, a set of *probe parameters* must be specified for every measurement before the external path planning can be executed. Probe parameters include the expected position, the probe orientation, the approach vector, the approach distance and the search distance. These parameters define the measurement and approach probes and are required to determine the location and path of the probe in a measurement cycle.

Many difficulties exist for adopting published methods for internal obstacle avoidance. Unlike most obstacle avoidance applications, such as mobile robot path planning, where the goal is simply to have the robot as far away from any obstacles as possible, the manipulator for **SuRP** is always in the neighborhood of the *obstacle* and controlled contact with the *object* is always required. The cell decomposition method would be time consuming and inefficient due to the complexity of the *object* and high degree of freedom of the probe. The potential method presents a problem of finding a valid potential function and describing potential surfaces in a three-dimensional setting for a sculptured surface. In addition, the *model* used to perform obstacle avoidance is an approximation of the *object*. A useful algorithm needs to be able to work with incomplete knowledge of the environment and satisfy the criteria of *robustness*, *efficiency* and *safety*.

A robust algorithm must work for a variety of surfaces, and it should guarantee a solution for any measurement on any surface, if such a solution exists. In this context, robustness for **SuRP** can be defined by equation 3.4.

$$\textbf{Robustness} = \frac{MEAS}{ATTEMPTS} \times 100\%. \tag{3.4}$$

$MEAS$ is the number of solutions predicited by the internal obstacle algorithm, and $ATTEMPTS$ refers to number of desired measurements. The efficiency of an internal obstacle avoidance algorithm can be defined in two ways: computational and measurement. A computational efficient algorithm measures efficiency of the obstacle avoidance algorithm in the traditional sense. Measurement efficiency measures the efficiency of the manipulator programs produced by the obstacle avoidance algorithms. A obstacle avoidance algorithm produces a list of commands for the manipulator, and measurement efficiency measures the efficiency of this list of commands. The optimization of measurement efficiency is usually built into the obstacle avoidance algorithm and is device and task dependent. For **SuRP**, one of the goals of internal obstacle avoidance is to find a probe orientation for every required measurement. Because each new orientation requires calibration for accuracy, the obvious goal of measurement efficiency is to minimize the number of probe orientations needed for a given list of measurements. The measurement efficiency for **SuRP** is defined by equation 3.5.

$$\textbf{MeasurmentEfficiency} = \frac{MEAS - ORI}{MEAS} \times 100\%. \tag{3.5}$$

where ORI is number of probe orientations used. Finally, a safe algorithm should produce the list of manipulator commands that is collision free. The safety of an algorithm is defined by equation 3.6.

$$\textbf{Safety} = \frac{MEAS - COLLISION}{MEAS} \times 100\%. \qquad (3.6)$$

where $COLLISION$ is the number of collisions.

There are tradeoffs between robustness vs. safety, and safety vs. efficiency, and the implemented internal obstacle avoidance algorithm attempts to balance these three criteria. For example, a conservative algorithm might produce probe parameters that are guaranteed to be collision free. However, the requirement for safety might constrain the algorithm such that it cannot determine valid probe parameters for most measurements. The search for a safe orientation might also require the algorithm to consider all possible orientations, and this search is obviously inefficient.

The implemented algorithm utilizes a conservative approach. It is an exhaustive search algorithm that is computationally inefficient but robust. The algorithm searches through a list of predefined orientations for measurement efficiency until a set of probe parameters satisfies a predetermined safety threshold to assure safety. The reason to use a conservative algorithm is simple: any collision might damage the object or the probe. Furthermore, because the overall method is iterative, any measurements that cannot be measured at one iteration is still available for measurement during the next iteration, due to the improvement in the understanding of the environment.

3.1.1 Search Strategy

The algorithm used is a hypothesize and test algorithm that uses the exhaustive search technique, and it is presented as a flow chart in Figure 3.2. The approach

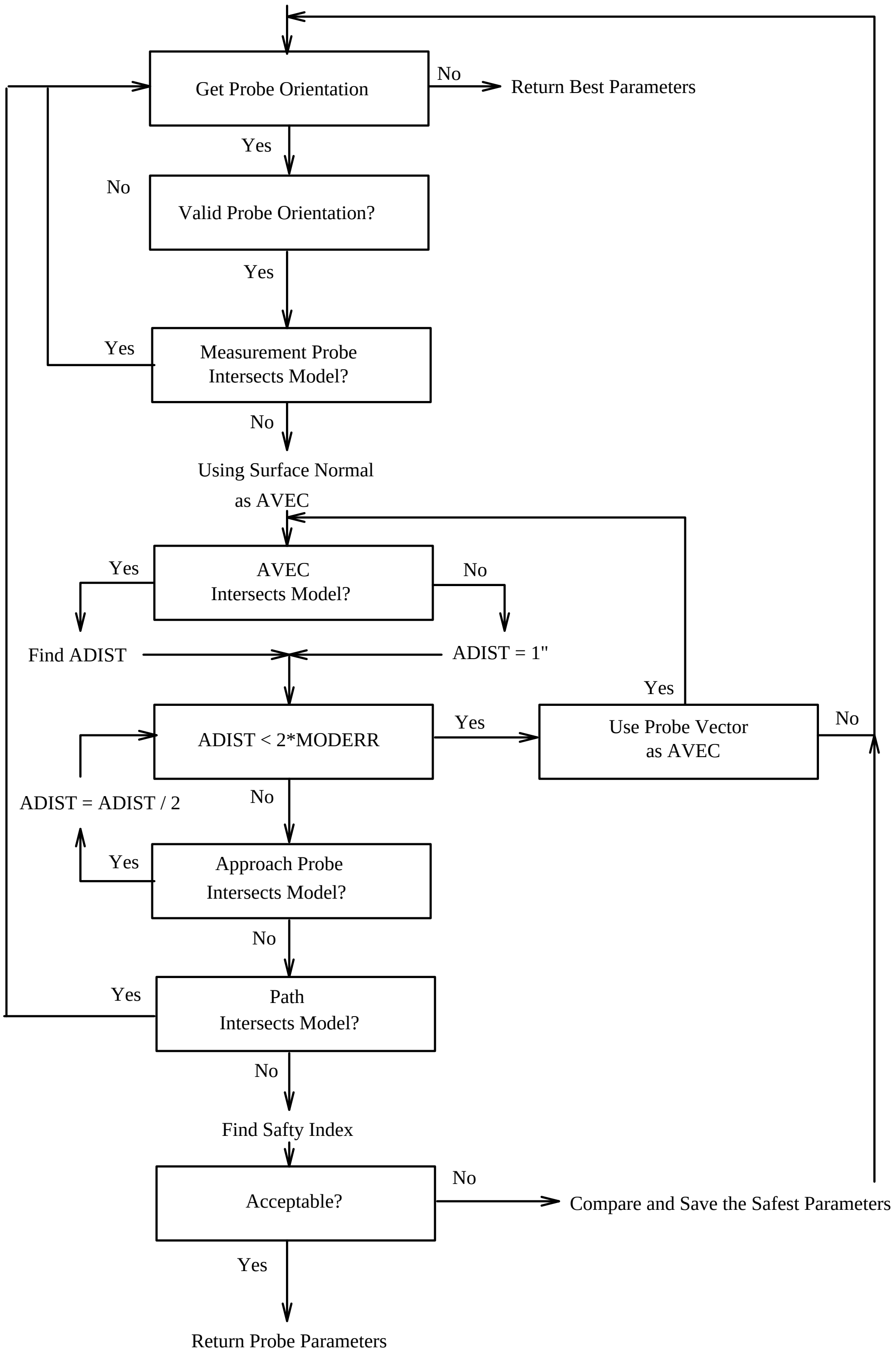

Figure 3.2. Flowchart for internal obstacle avoidance algorithm

might not be computationally efficient; however, it does satisfy the criteria of robustness, measurement efficiency, and safety. It achieves measurement efficiency and robustness by imposing a list of search orientations, beginning with primary orientation, then proceeds to secondary and so on. The primary orientations have roll and pitch angles that are multiples of 90°. Secondary orientations are multiples of 45°, and multiples of 30°, 15° and 7.5° are used for the remainder of the search list. With consideration of all possible orientations, robustness is achieved. The measurement efficiency is accomplished by searching in an orderly way, so that a few primary orientations can be used to measure most of the *object.* A set of rules is used to determine the validity and safety of each parameter being considered. A description of the algorithm is as follows:

1. The algorithm is given an expected position and the *model.* There are no best hypothesized approach probe and measurement probe.

2. The next probe orientation in the search list is retrieved. If the search list is exhausted, return the current best approach probe and measurement probe.

3. The validity of the probe orientation is determined by computing the *probe vector* of the probe orientation. Probe vector is a unit vector that represents the center line of the probe at the probe orientation (Figure 3.3).

   If the angle between the probe vector and the surface normal of the expected position is greater than 90°, the probe orientation is declared invalid, and the algorithm returns to step 2. An invalid probe orientation means that the probe with the probe orientation cannot make the correct contact at the expected position.

4. A measurement probe is generated with the probe orientation at the expected position. A surface-surface intersection test is performed between the measure-

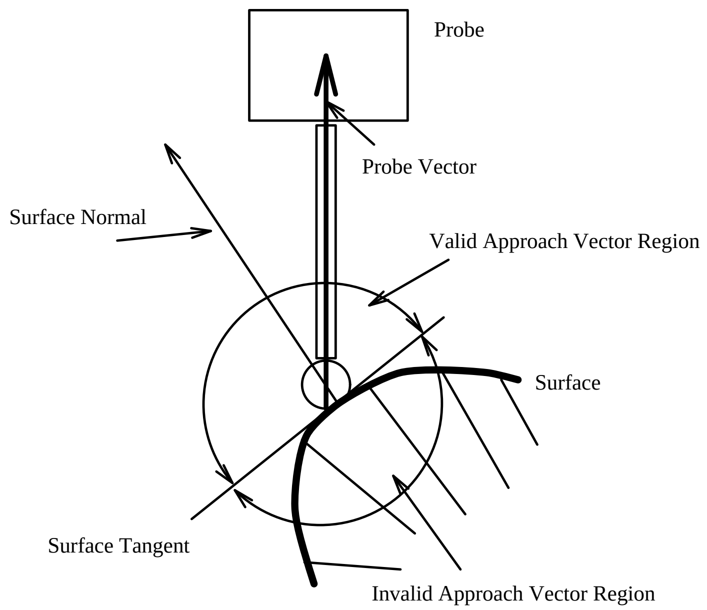

Figure 3.3. Illustration of probe vector and valid orientations

ment probe and the *model*. If the measurement probe intersects the *model*, the measurement probe is not safe, and the algorithm returns to step 2.

The surface-surface intersection test is described in section 3.1.2.

5. This step attempts to determine a valid approach position. In equation 3.1 an approach position is defined by an approach vector, an approach distance and an expected position. Because the expected position is given, the search narrows down to the approach vectors and approach distance. However, there are infinite combinations of approach vectors and approach distances; therefore, a set of rules is used to test a small subset of all possible vectors and distances.

   The algorithm limits the search of approach vectors to just two vectors. The preferred approach vector is the surface normal of the expected position. Due to the uncertainty of the *model*, any other approach vector might not produce the expected measurement. The other possible candidate as an approach vector is the probe vector defined in step 3. If the angle between the probe vector and the surface normal is less than 45°, the probe vector can be used as an approach vector when a valid approach position cannot be determined using the surface normal as the approach vector.

The reason for trying the probe vector is that it has a better probability of success than any other possible vector. In step 3 the probe vector is the vector in the probe orientation, and step 4 showed that a measurement probe is safe. Moving the probe in the direction of the probe vector involves traveling through the collision free space defined by the measurement probe; therefore, it is most likely that the probe vector could produce a valid approach position.

The following steps are used to find the valid approach vector and approach distance and illustrate in a two-dimensional example in Figure 3.4.

(a) Starting from the expected position, a ray is cast in the direction of the approach vector to determine the intersection between this ray and the

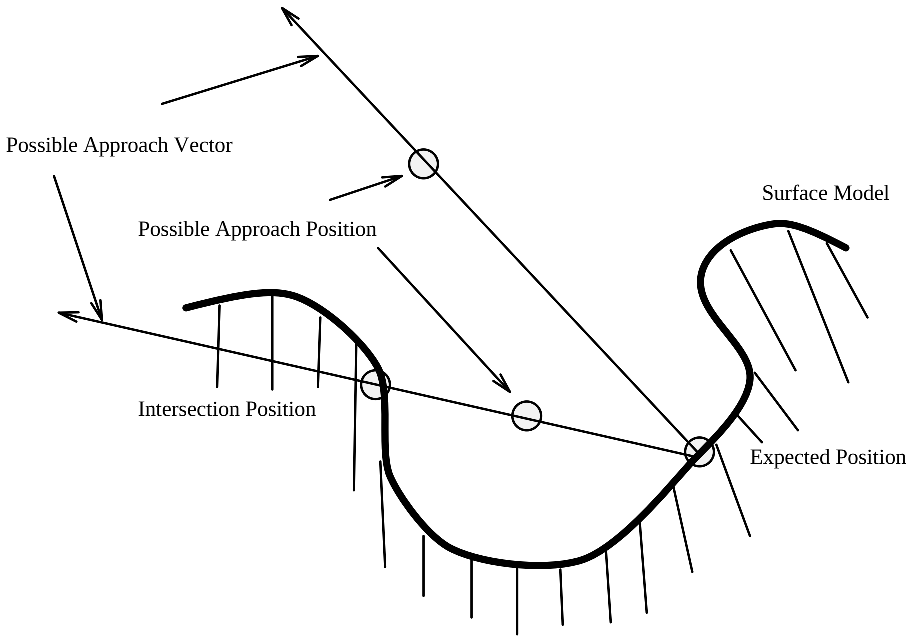

Figure 3.4. Locating valid approach position

model, using a ray-surface intersection test. If the ray and the model intersects, the approach distance is chosen to be the half of the distance between the expected position and the intersection. Otherwise, the approach distance is 1 inch by default.

If the approach distance is less than twice the *model error*, the approach probe might collide with the *object*, and the algorithm proceeds to step d. Model error represents the uncertainty of the *model* and is discussed in section 5.1

(b) An approach probe is generated at the approach position defined in step a. Surface-surface intersection test is performed between the approach probe and the *model*. If no intersection is reported, the algorithm proceeds to step 6.

(c) If the approach probe intersects the *model*, another approach position is proposed by halving the current approach distance and proceeding to step b.

(d) If the current approach vector is the surface normal of the expected position, the probe vector may be used as the approach vector and step a is followed. Otherwise, no valid approach position can be found for this probe orientation and the algorithm returns to step 2.

6. The path between the approach probe and the measurement probe needs to be verified to be collision free. This is done by generating *path probes* on the path and using surface-surface intersection test between each path probe and the *model*. If any path probe intersects the *model*, step 2 is returned to for a new hypothesis.

7. The *safety index* of the measurement and approach probe is collected. If the safety index is acceptable, the probe parameters are returned. Otherwise, the

safety index of the current probe parameters is compared with the stored best probe parameters, and the safer candidate is saved.

The safety index is collected using the surface-surface intersection test, and it is describe in section 3.1.3.

3.1.2 Surface-Surface Intersection

To efficiently and accurately compute surface-surface intersection is a difficult task. However, the goal of surface-surface intersection for **SuRP** is not to compute the intersecting curves between two surfaces but to determine whether or not two surfaces intersects. Furthermore, the *model* is an approximation of the *object*. It is acceptable to approximate the surface-surface intersection between the probe surface and the *model*.

The probe is approximated by four minimum bounding volumes, one bounding volume for each component of the probe: the stylus, the cylinder, the joint and the z-beam. The *model* is approximated by partitioning the entire surface into small surface patches, and the minimum bounding volume for each patch is used to approximate the surface. Intersection between bounding volumes of the probe and the surface can be easily computed by testing for bounding volume intersection between bounding volumes of probe components and surface patches. The resolution of the surface patch approximation is chosen to approximate the density of the measured data.

A linear search can be used to search for intersection through the entire list of surface patches. This search method, however, is extremely inefficient. A better search algorithm would be to utilize a tree search technique, and the surface patches are organized into a tree, similar to the *OCTTREE* encoding [28]. Neighboring patches are grouped and a larger patch is form by finding the union of smaller patches. This union patch becomes the parent node. Parent nodes can be further merged into grandparent node. The bounding volume of the entire surface is the root node of the tree. Figure 3.5 shows an example. Patches 1-9 are merged to produce parent nodes a-c, nodes a-c are merged to form node A and B. To search for intersection, each component of the probe is tested against the root node (ALL). If the bounding volume of the root node intersects the probe, its children (A and B) are tested for intersection. If there are no intersections between a node (A for instance) and the probe, then its children (node 1-6) do not need to be tested. The probe intersects the object if and only if the probe intersects a leaf node (1-9) at the end of the search.

Because the *model* used is an approximation, it is desirable to account for the error of the *model*. The intersection test uses bounding volume of the probe components. If the bounding volumes of the probe components are expanded on all sides by an additional distance of the *model* error, the algorithm will have accounted for the uncertainty of the *model*.

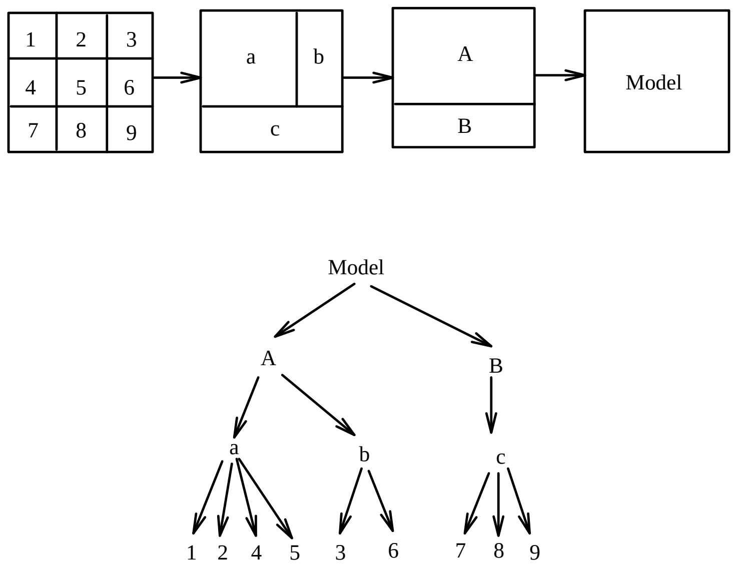

Figure 3.5. Data structure for surface intersection

3.1.3 Safety Index

The *safety index* is used to compute the safety of a measurement probe. Theoretically, the safety of the measurement probe can be defined as the minimum sum of the shortest distance from every point on the probe surface to the surface of the model.

Definition 1: *Let* $\vec{\mathbf{Pp}}(u_p, v_p)$ *be a point on the surface of the probe, and let* $\vec{\mathbf{Ps}}(u_s, v_s)$ *be a point on the surface of the model*

$$\mathbf{probesafety} = \sum_{u_p,v_p} min(dist(\vec{\mathbf{Pp}}(u_p, v_p), \vec{\mathbf{Ps}}(u_s, v_s)|\forall(u_s, v_s)). \qquad (3.7)$$

However, this is time consuming and difficult to compute. An alternative method is to use potential fields to estimate the safety of a probe by noting its location in the potential fields.

Potential fields can be estimated by finding the surface offset of the *model*. In other words, given a surface, expand the surface by some offset distance. However, large offset distance could create a potential surface with self-intersection, and it is impractical due to time and memory constraint to extend potential fields to cover the work space. Therefore, only one potential field is used in the **SuRP** implementation, and the safety of a probe is determined by a surface-surface intersection test between the probe and the potential field.

In section 3.1.2 surface-surface intersection is computed by finding intersections between surface patches and a probe. Using this approach, each intersecting patch means that for some regions on the probe surface, the distance between these regions and the model is smaller than the offset distance. More intersection patches mean that more regions on the probe are closer to the *model* than the offset distance of the potential field. The goal becomes finding a probe with a minimum number of intersecting patches with the potential field. Then the safety index can be defined as the number of intersections with the surface patches. A lower value of the safety index means a safer probe (Figure 3.6).

3.2 External Path Planning

External path planning answers the question of how to move the probe from one point to another outside of *obstacles*. Because the CMM is not capable of real-time manipulation and knowledge acquisation, it is assumed that **SuRP** has full knowledge of all *obstacles* in the work space. In most cases, the object to be measured, its supporting structures, and the calibration ball are the only *obstacles* presented in the CMM workspace. By modeling all *obstacles* as bounding boxes, the tasks of measuring *obstacles* are reduced to measure a few carefully selected points. Furthermore, this reduction in the complexity of *obstacle* representation results in

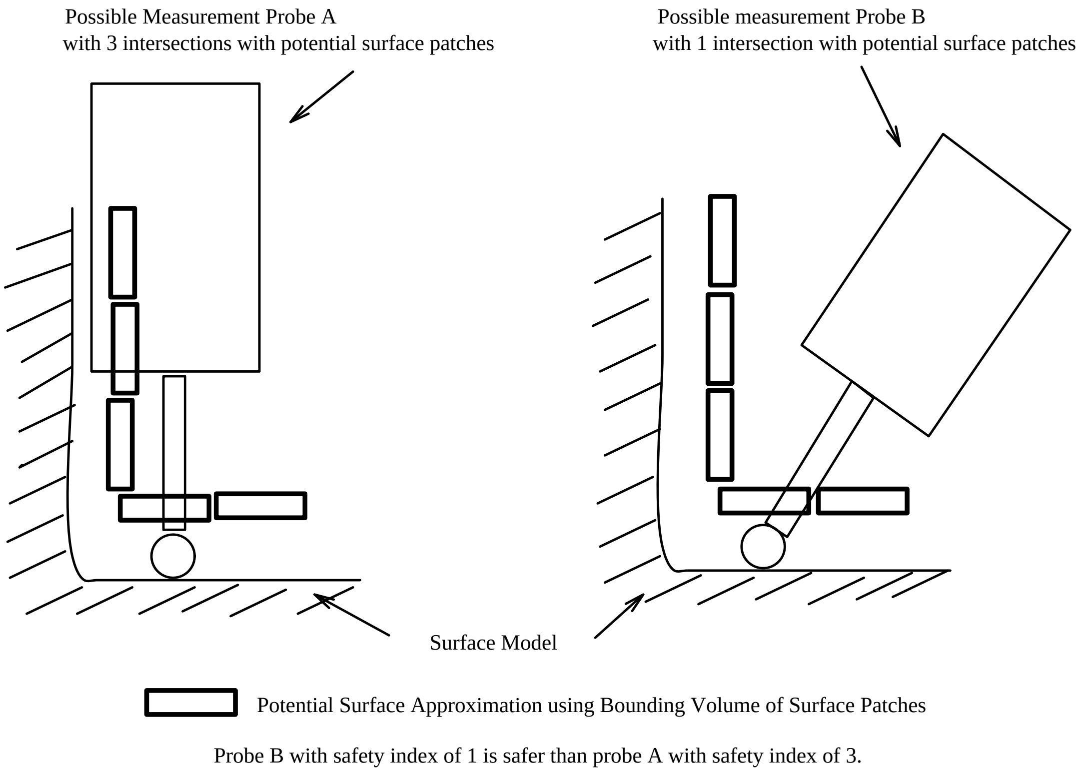

Figure 3.6. Illustration of safety index

an easy method to specify a tolerance distance of the obstacle, because the exact location and shape of obstacles are not known. Another consideration is that an efficient path in the CMM is a straight line. The delay of the CMM is noticeable whenever a change of direction is needed. Therefore, the optimal path for a CMM might not be the shortest path involving numerous changes of direction. Rather, the most efficient path could be the path with a minimum number of changes of directions.

With these considerations, the implementation for external path planning will utilize the free space method, because it is easy to partition the work space into a collection of bounding boxes, and the free space method guarantees a solution. The method of free space partitioning and the path search are described in section 3.2.1 and section 3.2.2.

3.2.1 Free Space Representation

The implementation for the external path planning will utilize the free space method, in particular, a method of coarse cell decomposition. In this scheme, *obstacles* and the probe with some specified probe orientation are approximated by bounding boxes, and the work space is decomposed into cells or boxes. By approximating the probe as bounding boxes, the physical size of the probe is taken into consideration. All bound boxes or cells have the same orientation, and for simplicity, they all align with the Cartesian coordinate system.

Cells are classified into three categories: safe, adjacent, and obstacle. An obstacle cell can either have the size and shape of the bounding box approximation of an *obstacle*, or it can be an adjacent cell that falls partly outside the work space or intersects other *obstacles*. Adjacent cells are adjacent to an obstacle cell. It has the size of the probe's bounding box. Within each adjacent cell, a position is designated the node position. The node position represents the stylus-ball position for the probe to occupy that adjacent cell. Finally, safe cells are spaces not designated as obstacle or adjacent cells, and they are disregarded in this implementation. Adjacent cells around an *obstacle* are constructed in the following manner and a two-dimensional example is illustrated in Figure 3.7.

1. None of the adjacent cells may intersect the *obstacle* they are adjacent to.

2. An adjacent cell may overlap other adjacent cells.

3. Any adjacent cell that intersects an *obstacle* that it is not adjacent to is categorized as an obstacle cell.

4. The faces of the *obstacle's* bounding box are classified into six groups: $xmin$, $ymin$, $zmin$, $xmax$, $ymax$ and $zmax$, according to its x, y, z intersect value. For example, if a bounding box has a face A with plane equation of $x = 10$ and face B with plane equation of $x = 20$, then face A is classified as $xmin$ and face B is $xmax$.

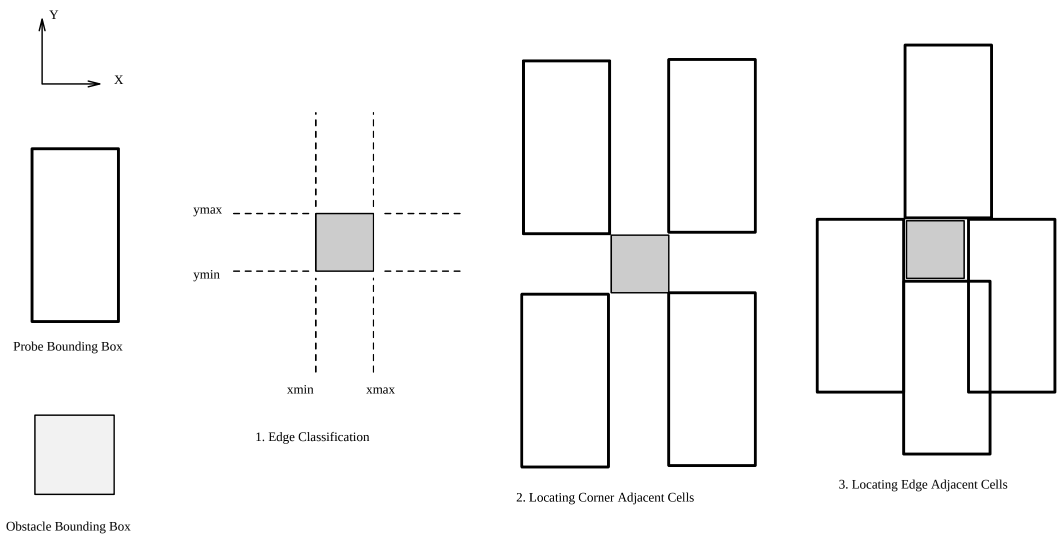

Figure 3.7. Construction of adjacent cells around an *obstacle*

5. Corner adjacent cells are created and placed into the corners partitioned by the six faces of the *obstacle's* bounding box. It must only share a corner with its adjacent *obstacle*.

6. Edge adjacent cells are created in a manner such that an edge adjacent cell shares at least one face with a corner adjacent cell or another edge adjacent cell, and it must also share an edge with the *obstacle's* bounding box. All the space along the edge of the *obstacle* must be classified either as an adjacent or obstacle cell.

7. Face adjacent cells are created in a manner such that a face adjacent cell shares at least one face with an edge adjacent cell or another face adjacent cell, and it must always shares a face with the *obstacle's* bounding box.

To move from one adjacent cell to another, the probe is constrained so that it may only move from one adjacent cell to another if and only if these two adjacent cells share the same face. This scheme constrains the movements from one adjacent cell to another in only six directions, $+/-X$, $+/-Y$ and $+/-Z$. The resultant adjacent cells completely surrounds their *obstacle*. This is desirable, because the manipulator spends most of its time in the vicinity of the *object*, which is also an *obstacle*. Adjacent cells also act as a buffer so that any time a probe intersects an adjacent cell, it is forced to move in the manner described in section 3.2.2.

Figure 3.8 shows an example of the coarse cell decomposition scheme in two dimensions with three obstacles. One drawback of this technique is that it forces us to recompute the decomposition for every probe orientation. However, since there are only a few cells to consider and all cells are boxes aligned in the XYZ coordinate frame, it is relatively fast to perform this operation.

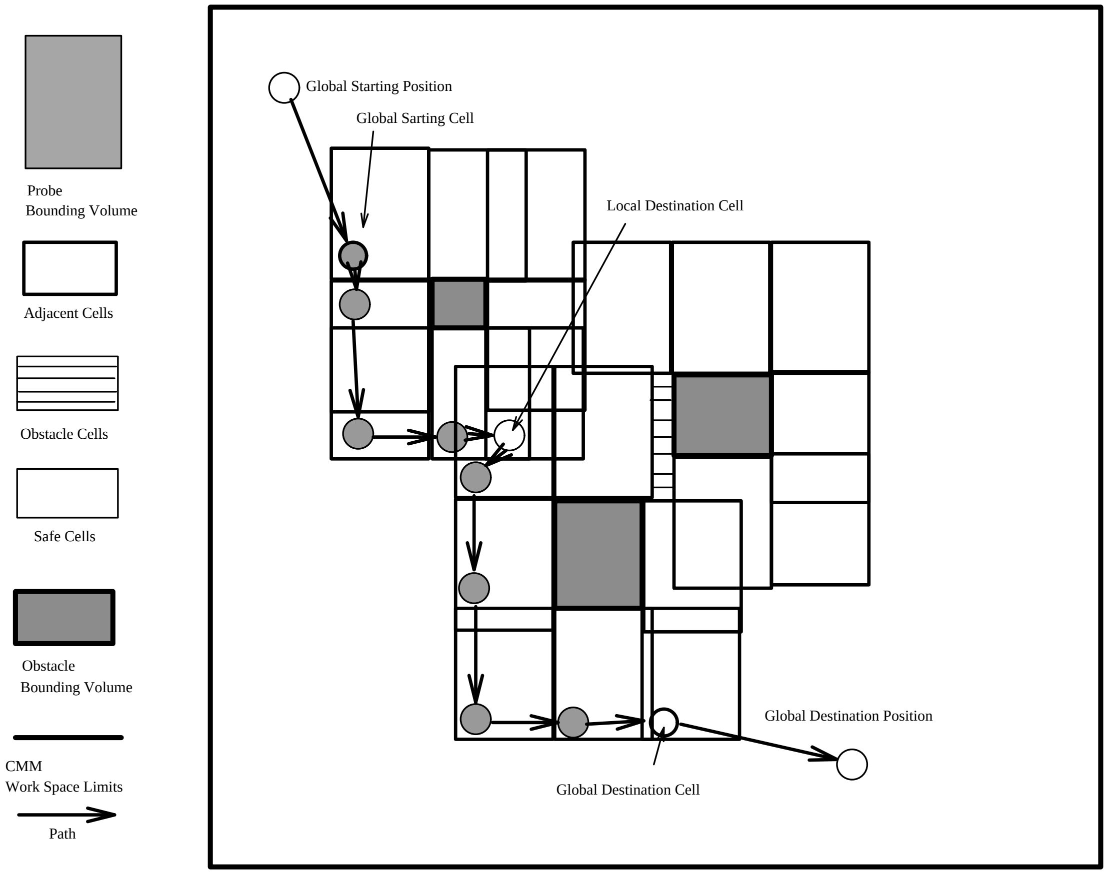

Figure 3.8. An example of coarse cell decomposition

3.2.2 Path Search

Once the free space is decomposed, it is possible to use a variety of techniques to search for an optimal path. The implemented method reduces the problem of path planning into finding collision free path for a single obstacle. The method works by moving around one obstacle at a time until the destination is reached. An example of the path search is shown in Figure 3.8. A description of the algorithm follows,

1. Given the global-starting position and global-destination position, and the decomposed free space.

2. Find the closest adjacent cell to the global-starting position with a collision free path. To determine the collision free path between two cells, the union of

these two cells are checked to make sure they do not intersect any obstacle cells. The closest adjacent cell to the global-starting position is the global-starting cell ($GSTART$). The same operation is used to find the global-destination cell ($GDEST$) using the global-destination position.

3. If the $GSTART$ and the $GDEST$ are adjacent to the same obstacle, the single obstacle avoidance is performed. This operation is described later. If $GSTART$ and $GDEST$ are the same, a path has been found, and the algorithm returns all stored intermediate cell locations.

4. If the $GSTART$ and the $GDEST$ are not adjacent to the same obstacle, a local-destination cell ($LDEST$) adjacent to the same obstacle as the $GSTART$ is determined using the same operation outlined in step 2. The single obstacle avoidance is performed between $GSTART$ and $LDEST$.

5. Set $GSTART$ equal to $LDEST$, and go to step 3.

Single obstacle avoidance is a simple way to move the probe bounding volume around one obstacle cell, through a sequence of moving from one adjacent cell to the another adjacent cell. The allowed movement from one adjacent cell to another is restricted to 6 directions, $+/-X$, $+/-Y$, and $+/-Z$. The algorithm is described as follows:

1. A starting adjacent cell (SAC) and a destination adjacent cell (DAC) is given.

2. If SAC equals to DAC, the list of intermediate cells are returned.

3. The differences in SAC and DAC are computed. The difference is described by the six direction component. For example, if the Cartesian coordinate of the DAC is (20,10,10), and the SAC coordinate is (10,20,10), then the differences are $+X$ and $-Y$.

4. For one direction difference, if the adjacent cell in this direction of the SAC is free (i.e., it is not an obstacle, and it has not been visited previously), the probe moves to this cell, by setting SAC to this cell location, and step 2 is then followed.

5. If this adjacent cell is not free, another direction difference is chosen.

6. If the cell cannot be advanced after trying all directional differences, an attempt to advance in the direction of a similar direction is tried. Using the same example, similar directions are $-X$, $+Y$, and $+/-Z$.

7. When all the similar directions are tried and it is still not possible to advance, the algorithm reports failure.

3.3 Data Acquisition Algorithm

Once both components of the data acquisition method are defined, a method is needed to link them together. Because the internal obstacle avoidance algorithm finds a collision free path within the bounding box representation of the *obstacle*, and external path planning finds a path outside the bounding boxes, a method is required to find a collision free transition in and out of the bounding boxes.

For a desired measurement, the data acquisition algorithm first uses the internal obstacle avoidance routine to find a safe path between the approach probe and the measurement probe. If the approach probe falls entirely outside the *object's* bounding box, this position can be used as a destination position for external path planning. However, if the approach probe falls within the *object's* bounding box, an *adjacent probe* needs to be determined. The adjacent probe defines a position in the adjacent cells that the approach probe can safely go to in a linear path. To find such a path, an algorithm similar to step 5 of the internal obstacle avoidance routine is used. The algorithm assumes that the probe vector can provide a safe path. A position outside of the bounding box using the probe vector can be determined and an adjacent probe is hypothesized. Using the path-surface intersection algorithm presented in step 6 in section 3.1.1, the transitional path is verified. If no safe transition path can be found, another set of probe parameters for this measurement will need to be determined. After adjacent probes have been defined for all measurements, external path planning is utilized to move the probe from one adjacent probe location to another.

CHAPTER 4

SURFACE ESTIMATION

In vision and object recognition, the surface representation problem has the benefit of dense data [29, 30, 31, 32]. However, noise and data reduction are as much of an issue as surface fitting. Data acquired by the CMM are sparse and scattered but accurate; therefore, it seems that approximation or interpolation techniques from computer graphics literatures [33, 34] might be appropriate. In their surface reconstruction system using a CMM, Lee et al. [17] essentially used a least squares method to approximate their data.

For simplicity, this system will use a B-spline approximation package from the Alpha_1 modeling system [3]. Interpolation techniques might be ideal, because real measured points are available; however, the shapes of interpolations are inherently unstable and hard to control.

B-spline surface can be formulated by equation 4.1.

$$\vec{\mathbf{Ps}}(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} \vec{\mathbf{P}}_{j,k} N_{j,s}(u) N_{k,t}(v). \tag{4.1}$$

The Cartesian product of the blending function $N_{j,s}$ and $N_{k,t}$ defines a parametric vector function over the surface. $\vec{\mathbf{P}}_{j,k}$ defines the $(m+1)$ by $(n+1)$ control points. S and t are the polynomial degrees used for the blending function.

The measured points are used to approximate a set of curves, and these curves are used as control curves to create an approximated surface. Because control curves need not be planar, and they do not require same number of control points, this approach is more flexible than the traditional B-spline approximation method.

Given enough points, the approximated surface will converge to interpolating the data points.

When a *model* is accepted as final, the *model* computed needs to be corrected for probe error. This is accomplished by moving all the points on the surface in the opposite direction of their surface normal by the distance of the radius of the stylus ball.

4.1 Hierarchical Approximation

One difficult aspect of using an approximation for surface fitting is that the approximated surface usually does not pass through data points, which is true for the B-spline method used here when there is sparse data. Because the approximated surface is used to predict obstacle avoidance and next iteration point sampling, a hierachical approximation technique is implemented to avoid poor approximations.

B-spline curves and surfaces can be controlled by specifying polynomial degrees of the functions used for the approximation. A first order approximation would produce linear curves or bilinear surfaces passing through every control point. With each successively larger order, the approximate curve or surface falls further away from the control points.

In the hierarchical approximation approach, the data are first approximated using cubic B-spline functions. If the distance between the data points and the approximation is greater than some specified distance criterion, the order of the approximation is reduced by one. This process is iterated until the distance criteria is satisfied or the first order approximation is used.

4.2 Data Organization

Another problem with using the B-spline approximation for surface fitting is that a rectangular control mesh is required. A control mesh is a two-dimensional matrix of control points. If measured points are used as control points, it is not possible to measure just the points selected to refine the *model*. For an m by n control mesh, $O(m+n)$ points need to be measured for one desired point in order to maintain the control mesh. Furthermore, geometric order of the control points in the mesh needs to be maintained; otherwise, the approximation will behave unpredictably.

For this reason, *Alpha_1 control curves* [3] are used. Control curves are B-spline curves. In this scheme, data points are combined into a number of control curves. These control curves are used to approximate the surface. Each control curve need not have the same number of control points. Geometric order of the control points is also kept strictly within the control curve and among the control curves. Let m be the number of control curves and n be the maximum number of control points on a control curve. Adding a new point means that in the worst case scenerio, n points are going to be measured. Another advantage of using control curves is that when some required data are not measured, the control curve method can still produce an approximation.

To create a rectangular control mesh from a set of control curves, n is first determined. For control curves with less than n control points, additional control points can be estimated and the control mesh is filled to maintain the rectangular shape.

CHAPTER 5

EVALUATION AND VERIFICATION

Once the *model* has been reconstructed using the measured data, it is necessary to consider how well this model fits the data and how well the *model* compares to the original object. The second question is an inspection problem and is more difficult to answer, because we do not have a priori knowledge of the *object*. Even in inspection tasks, where a CAD/CAM description of the surface is known, the task of verification is under research investigation [13, 15]. In our application, we can answer how well our approximated surface corresponds to the measured data. This is presented in section 5.1.1. A natural followup to this quesion is how can the *model* be improved. This is discussed in section 5.1.2.

Another parameter to consider in performance evaluation is the CMM. There are a variety of factors that could contribute to the performance of the machine. Elshennawy et al. [35] provided a list of common factors: geometric errors, thermal distortion, kinematic errors, static and dynamic errors, work piece errors and probe-work piece interaction. All of these factors are beyond the scope of this research. However, steps can be taken to reduce these effects. For example, the CMM can be installed in a controlled environment to reduced the effect of temperature change and each probe orientation can be calibrated to ensure accuracy. Also, the object can be mounted on a stiff base to reduce errors caused by deflection and vibration.

A last consideration is errors caused by the measurement program produced by the **SuRP**. This is discussed in section 5.2.1.

5.1 Evaluation

The proposed process is iterative. The concept of iteration is to refine the *model* at each iteration by gathering more data in the regions of largest discrepancies. Therefore, a method and criteria are needed to evaluate the *model* with respect to the measured data.

One obvious criteria for evaluation is the Euclidean distance between data points and the surface. A mapping between measured points and the estimated surface is accomplished; that is, we determine a closest point on the surface to the control point. The distance between this estimated position and the control point is considered as *position error*(**poserr**).

$$\mathbf{poserr}(r, c) = distance(\vec{\mathbf{P}}(r, c), \vec{\mathbf{P}}\mathbf{s}(u(r), v(r, c))) \tag{5.1}$$

where r and c are the parameters of the control point, and $u()$ and $v()$ are the mapping functions. Function $\vec{\mathbf{P}}(r, c)$ is used to find the position vector of a data point from the original data set. This function returns the position of the c-th control point on the r-th control curve. The range of c depends on the value of r, because not all control curves have the same number of control points. Mapping functions are defined in section 5.1.1.

If **poserr** of a point is beyond a certain threshold, the neighborhood surrounding this point is measured. If all position errors are within some specified threshold or converge, the system terminates and the surface produced is considered the final result.

Model error (**moderr**) is defined as the largest position error for the entire set of data points in the *model*. **Moderr** determines the uncertainty of the *model*.

$$\mathbf{moderr} = max\{\mathbf{poserr}(r, c) | \forall(r, c)\}. \tag{5.2}$$

5.1.1 Mapping

Because the distance between the *model* and measured data points is used as a criterion for evaluating the goodness of the surface, a robust method is needed to create a mapping between the measured points and surface points. In other words, for every data point $\vec{\mathbf{P}}(r,c)$, the goal is to find functions, $U(r)$ and $V(r,c)$, such that $distance(\vec{\mathbf{P}}(r,c), \vec{\mathbf{P}}\mathbf{s}(U(r), V(r,c)))$ is minimized, subject to the geometric ordering constraint.

Definition 2: *$U(r)$ and $V(r,c)$ maps $\vec{\mathbf{P}}(r,c)$ to the model, $\vec{\mathbf{P}}\mathbf{s}(U(r), V(r,c))$,*

if and only if

$$distance(\vec{\mathbf{P}}(r,c), \vec{\mathbf{P}}\mathbf{s}(U(r), V(r,c))) < distance(\vec{\mathbf{P}}(r,c), \vec{\mathbf{P}}\mathbf{s}(Ux(r), Vx(r,c)))$$

$$\forall Ux() \neq U() \text{ and } Vx() \neq V() \text{ and}$$

$$U(), Ux() \in [umin, umax], \text{ and } V(), Vx() \in [vmin, vmax],$$

and

$$\text{if } r1 < r2 \text{ and } c1 < c2, \text{ then } U(r1) < U(r2) \text{ and } V(r1, c1) < V(r1, c2).$$

If the surface has an open-ended condition in both u and v direction, then the mapping between corner data points is simply the corner points of the surface (e.g., $\vec{\mathbf{P}}(0,0)$ maps to $\vec{\mathbf{P}}\mathbf{s}(umin, vmin)$). Because the data points on the edge must also reside on the boundary of the surface, the first column of every row ($\vec{\mathbf{P}}(r,\cdot)$) must map to the left edge ($v = vmin$) of the surface, and a search for the mapping is performed. In other words, all $\vec{\mathbf{P}}(r,0)$ maps to $\vec{\mathbf{P}}\mathbf{s}(U(r), vmin)$.

Because each row is approximated by a curve, all the points on the same row must share the same u as the first point. In other words, $U(r)$ is constant for all data points on r-th control curve, $\vec{\mathbf{P}}(r,\cdot)$. The search space is limited again to a single curve for each row. If the surface has periodic condition, the corner points are localized first by searching in the window centered around the corner points of the surface.

The mapping algorithm searches in a small partition of the curve for a point on the curve that minimizes the distance between the two points. The size of the partition decreases as the mapping is localized. A detail description of the algorithm follows:

1. First, the algorithm finds an expected parametric value (EPV) and searches window size. If n is the number of data points on r-th control curve, then for c-th points, the EPV is compute as

$$EPV(c) = vmin + \frac{vmax - vmin}{n} \times c. \tag{5.3}$$

   If the curve is open-ended, the window size is zero for the first and last point. Otherwise, the window size is

$$windowsize = \frac{vmax - vmin}{n}. \tag{5.4}$$

2. If the window size is too small, the algorithm terminates, and

$$V(r,c) = EPV(c) \tag{5.5}$$

   and

$$\mathbf{poserr}(r,c) = D(c). \tag{5.6}$$

   where $D(c)$ is the distance between $\vec{\mathbf{P}}(r,c)$ and $\vec{\mathbf{Ps}}(U(r), EPV(c))$.

   $Dr(c)$ is the distance between $\vec{\mathbf{P}}(r,c)$ and $\vec{\mathbf{Ps}}(U(r), EPV(c) + windowsize)$.

   $Dl(c)$ is the distance between $\vec{\mathbf{P}}(r,c)$ and $\vec{\mathbf{Ps}}(U(r), EPV(c) - windowsize)$.

3. If $D(c)$ is less than or equal to both $Dr(c)$ and $Dl(c)$, $windowsize$ is reduced by a half and step 2 is repeated.

4. If $Dr(c)$ is less then $D(c)$, $EPV(c)$ is set to $EPV(c) + windowsize$ and step 2 is repeated.

5. If $Dl(c)$ is less then $D(c)$, $EPV(c)$ is set to $EPV(c) - windowsize$ and step 2 is repeated.

The result of this mapping is a grid of data. Each value on the grid represents the error of a control point relative to the surface. A glance at this grid provides an immediate feedback to the distribution of errors and can be used to devise a sampling plan.

5.1.2 Sampling Plan

Once the error mapping is produced, a plan is needed to determine the locations for measurements to improve the *model*. The goal is to identify the defective points that are data points with large position errors. More data are measured in the vicinity of these defective points.

A sampling plan must work with the surface approximation method presented in section 4.2. In section 4.2 the surface is approximated using control curves. The approximated surface cannot be better than the approximated curves used. Therefore, the source of error on a surface could be due to two factors:

1. *Curve point error* (**cpe**) are errors caused by bad approximation of a control curve due to a lack of control points.

2. *Surface point error* (**spe**) are errors due to bad approxmation of a surface due to a lack of control curves.

First, the tolerance of the acceptable *model* needs to be specified by the operator. This *model tolerance* (**modtol**) is used to determine new measurements and the termination of the process. Next, a *surface error map* (**sem**) between the data points and the surface model is produced, and the *curve error maps* (**cem**) between data points and the control curves are also produced. **Sem** and **cem** are produced using the algorithm described in section 5.1.1. With the model tolerance and the error maps, a list of *defective points* can be determined.

Definition 3: $\vec{\mathbf{P}}(r,c)$ *is defective, if and only if*

$$\mathbf{cem}(r,c) > \mathbf{modtol}$$

or

$$\mathbf{sem}(r,c) > \mathbf{modtol}.$$

Definition 4: *Defective point* $\mathbf{P}(\vec{\mathbf{r}},\mathbf{c})$ *is* **cpe***, if and only if*

$$\mathbf{cem}(r,c) > \frac{\mathbf{sem}(r,c)}{2}.$$

Otherwise,

$$\mathbf{P}(\vec{\mathbf{r}},\mathbf{c}) \textit{ is } \mathbf{spe}.$$

Every defective point is remeasured to make sure the error is not caused by a faulty CMIS program. For each defective point, the error is classified into **cpe** or **spe**. If a defective point whose error in the curve error map is at least half the error in the surface error map, this defective point is categorized as **cpe**; otherwise, it is **spe**. If a defective point is classified as **cpe**, then additional points are sampled on the control curve of the defective point. For every defective point with **cpe**, new points are measured and inserted halfway between the defective point and its neighboring points. However, if a defective point is classified as **spe**, a new control curve is measured between control curve A and B if and only if

1. both curve A and B have no defective points with **cpe**,

2. at least one defective point with **spe** on A or B,

3. curve C, a control curve adjacent to the curve with **spe** defective points, also has no defective points with **cpe**.

If a new control curve is to be added between curve A and B, then the new curve would have the same number of points as curve A or B, whichever is greater.

5.2 Verification

One aspect of verification is the verification of measured data to ensure their accuracy, and all measured data needs to be verified before it can be combined with old data. Another aspect of verification is to verify the statistical error measure of the *model*. Because the new measurements are extracted from a *model* with some expected model error, the position error between the expected position and the measurement position should fall within some expected value. These two verification issues are described in section 5.2.1 and section 5.2.2.

5.2.1 Data Verification

Even with all of the precautions, it is possible that the measurement programs produced by the system will make mistakes, especially in the early iterations when the knowledge of the *object* is poor. There are three common type of errors: *missed measurements*, *collision* and *mis-measurements*.

For missed measurements, the measurement program attempts to measure a point on the *object* that does not exist; that is, the estimated position from the *model* to be measured is not on the *object*. This is usually caused by poor approximation. As mentioned in section 3, when the CMM is unsuccessful, the probe proceeds to the end of the search and waits. When this occurs, the probe must be triggered manually and the CMIS program would continue. Because the probe is triggered at the end of the search distance, if the distance between the measured and expected position is approximately the same as the search distance, these erroneous measurements can be detected and deleted by comparing measured and expected data.

The other possibility is collision between the probe and an *obstacle*. If the *obstacle* is securely fixed, the CMM system would crash, and the offending statement could be deleted to allow for completion of the measurement program, and data is never measured. However, if the probe was able to move the *object*, the process needs to be restarted from the beginning. It is possible to obtain some registration points on the original object and the replaced object, and compute the transformation needed [36, 37]. The old model can be transformed such that it will align with the replaced object. However, this transformation would introduce new errors, and the old data might not be compatible with the measurements taken from the replaced object. To acquire a good model, the replaced object would still need to be remeasured.

Finally, mis-measurement occurs when the sensor is triggered prematurely, primarily when there is an unexpected collision between the stylus component of the probe and the object. This error can be treated the same way as the missed measurements. When comparing the expected and measured positions, the data are discarded if the distance between the two is greater than some expected tolerance. The expected tolerance is set to equal to twice the model error computed from 5.1.1. This value is chosen for two reasons:

1. the *object* might be +/- model error unit away from the model, and

2. the extra tolerance is allowed for possibility of a faulty *model*.

Some mis-measured data might be acceptable. If this happens, then we can expect the model to have some large error, and the mis-measured data would be classified as defective points. Because defective points are remeasured, the mis-measured data could be corrected at the future iteration.

Because the search distances are usually greater than the model error, a simple condition can be used to verify the measured data.

Definition 5: *A measured data point is valid if*

$$distance(\mathbf{me\vec{a}pos}, \mathbf{ep\vec{o}s}) \leq \mathbf{moderr} \times 2.0. \tag{5.7}$$

Another cause of mis-measurement is the inaccurate model. An inaccurate model may cause a measurement data to result in crossover. This error is harder to detect, and currently, operator intervention is required. Therefore, the criterion in equation 5.7 is only a rule of thumb. Because the process is integrated into a geometric modeler, the user may be required to intervene to assure accuracy and validity of the measured data.

5.2.2 Model Verification

Once the measurements are verified, *model* verification can be considered. The concept involves using the new measurements to verify the *model* from the old data set. For instance, at iteration n, the model error computed is determined to be errmax; the measurements using this *model* are expected to have errors of less than errmax. If all errors are less than errmax, and errmax is less than the predetermined acceptable tolerance level, **modtol**, and the new *model* also has model error of less than **modtol**, then the model is acceptable. Otherwise, at least one more iteration is needed.

However, *Nyquist sampling theorem* [38] states that in order to recover a signal, the *Nyquist rate*, the sampling frequency, must exceed twice the *Nyquist frequency* of the signal. In [39], Marvasti et al. presented a paper on signal recovery using nonuniform samples and iterative methods. The problem and the approach are similar to surface reconstrution issues presented in this thesis. However, in their algorithm, they assume the Nyquist frequency of the original signal is known. Because the Nyquist frequency of the *object* is not known and would be difficult to acquire, it is possible that the accepted model might not recover some high frequency components of the *object*. One way to prevent this error is to make sure the initial *model* includes the high frequency component. The other method is to sample points randomly on the accepted *model* and the *object*. These points are verified to ensure the accuracy of the accepted model. If the randomly sampled points show unacceptable errors, more iterations can be performed.

CHAPTER 6

EXPERIMENTAL RESULTS

Experiments of the **SuRP** was performed on a variety of objects: a sculptured pocket, compressor disk blade surface, a plastic model of a human vertebra and a plastic model of a human femur. Experimental results of these objects are presented in section 6.1, section 6.2, section 6.3 and section 6.4. Performance evaluation of the **SuRP** is presented in section 6.5.

All computations were performed on a Sun Sparcstation 2. The COMERO CMM with a Rennishaw PH10 touch probe was used to execute the CMIS programs. A stylus with length of 42 millimeters (1.6540 inches) and diameter of 2 millimeters (0.0785 inches) attached to the probe is used for all experiments. This probe configuration is capable of meauring the surfaces on all experimental objects.

6.1 Sculptured Pocket

The sculptured pocket object was designed and manufactured using the Alpha_1 modeler [3] and a three-axis milling machine. The original object is shown in Figure 6.1.

The entire top surface including the sculptured pocket was subjected to the reconstruction process. The top surface has surface area of approximately 14.3 square inches. Initially, 16 points were measured on three control curves. The first order control curves are shown in Figure 6.2, and Figure 6.3 shows the shaded bi-linear surface of the initial model. The initial model has maximum error of 0.6121 inches using a third order B-spline surface approximation.

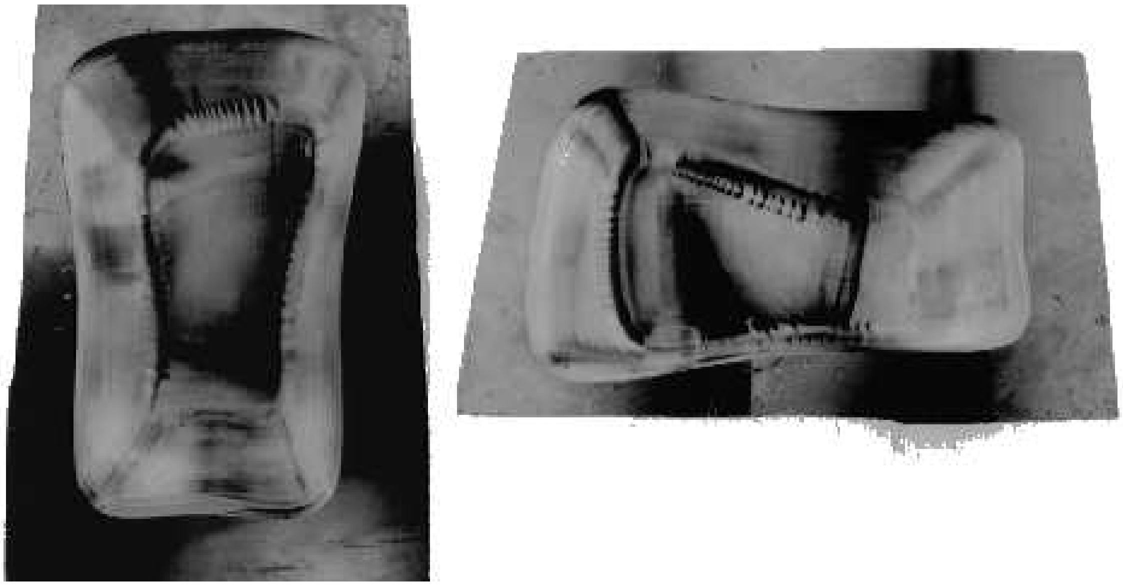

Figure 6.1. Original sculptured pocket object

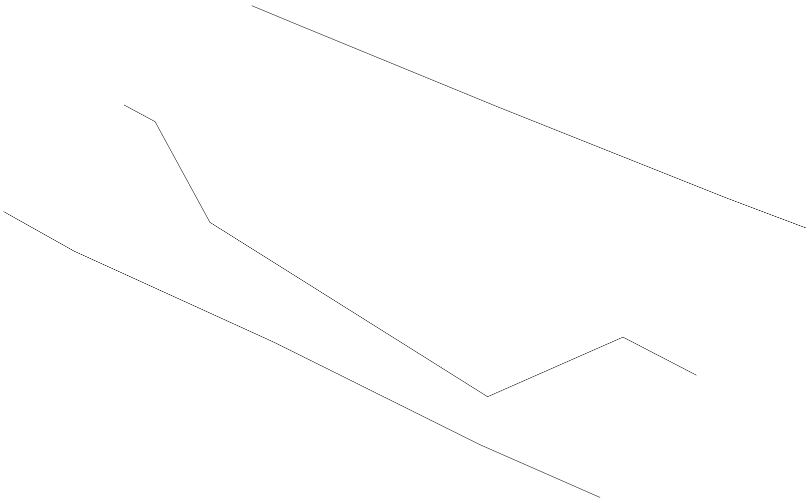

Figure 6.2. Initial control curves and control points for the sculptured pocket object

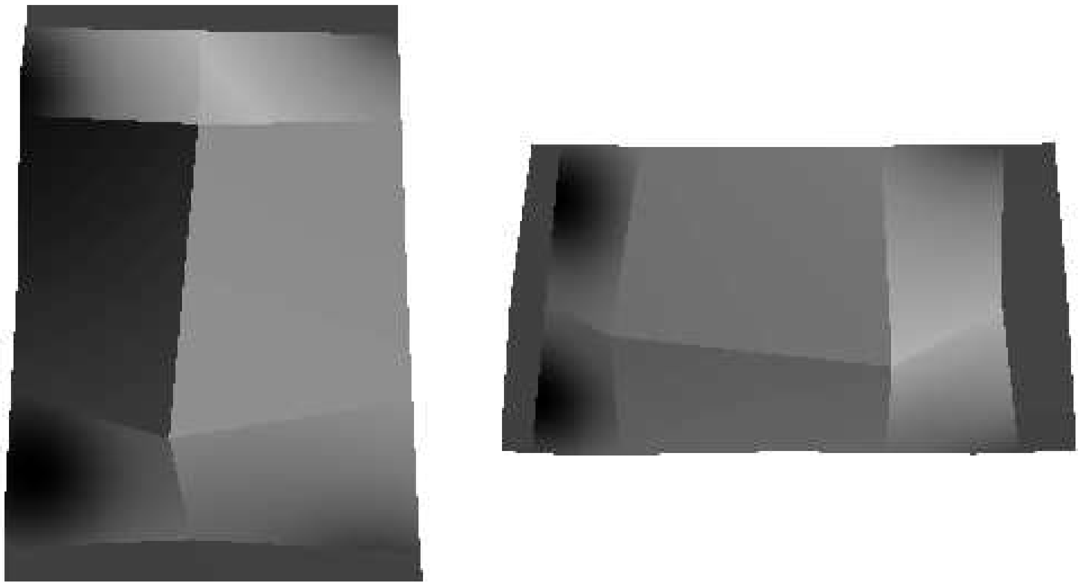

Figure 6.3. Two views of the initial sculptured pocket model

Figure 6.4 shows the number of data points used to model the sculptured pocket object at each iteration, and Figure 6.5 shows a plot of the maximum and average error at each iteration. These errors are computed using a third order B-spline approximation. In addition, the maximum error of new measurements at each iteration is also shown. The new measurement errors are position error between the expected and measured position. The new measurement errors show the confidence of the model error at each iteration. If the new measurement error is greater than the maximum error, then the maximum error is not a good estimate of the model error, and more iterations might be required. On the other hand, if the new measurement error is less than the maximum error, maximum error may be a good measure of the model error.

The efficiency measures of the internal obstacle avoidance algorithm for every iteration is shown in Figure 6.6. These efficiencies were discussed in section 3.1.

It took 14 iterations and about 8 hours to produce the final model. The maximum model error of the final result is 0.0313 inches, with 90% of data points having errors of less than 0.0149 inches. The density of the final data is about 41

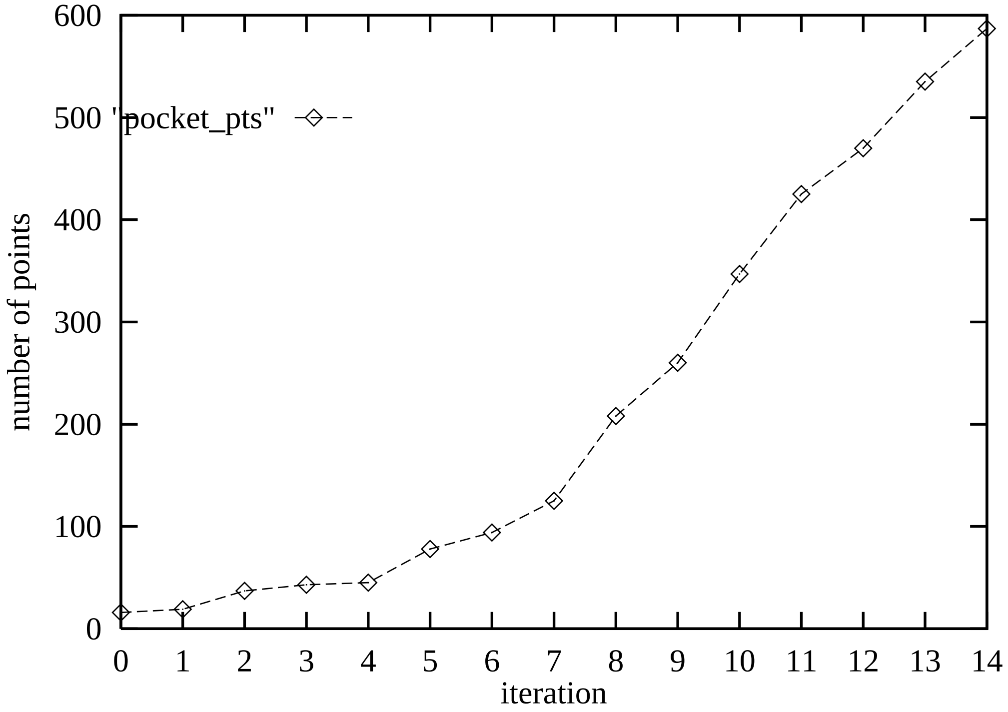

Figure 6.4. Plot of number of points at iteration for the sculptured pocket object

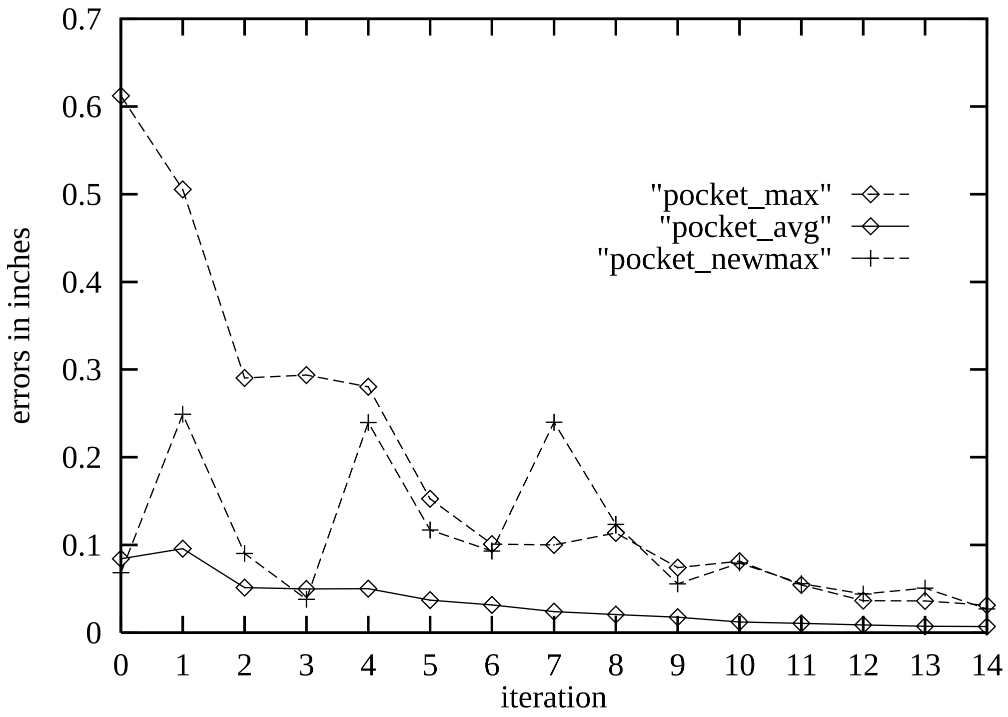

Figure 6.5. Plot of maximum and average error vs. iteration for the sculptured pocket object

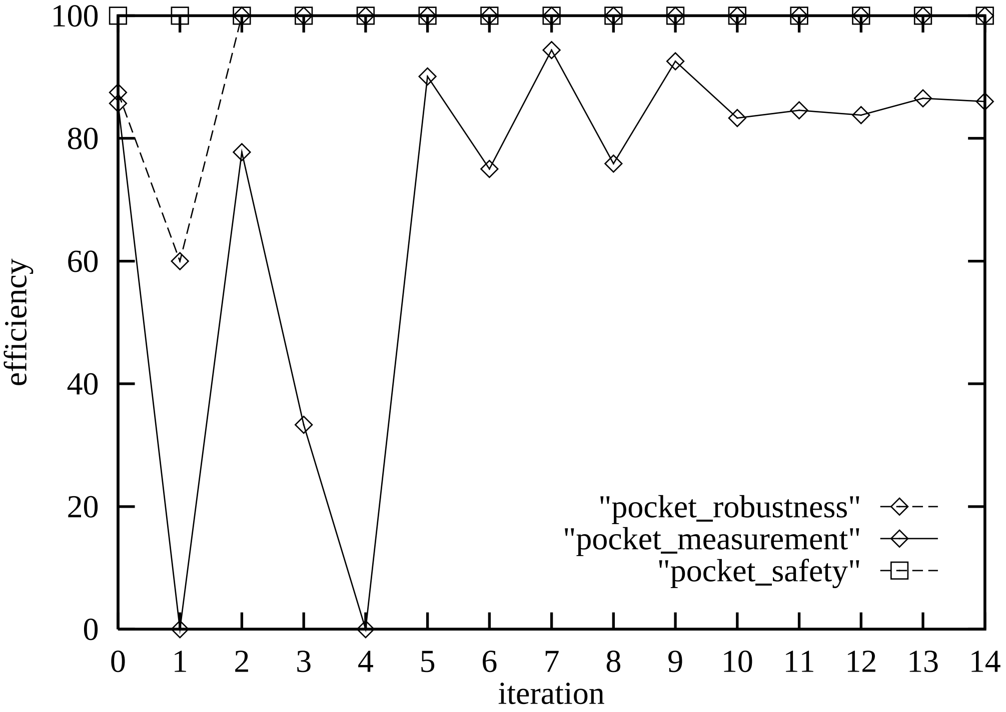

Figure 6.6. Plot of efficiency of the internal obstacle avoidance algorithm for the sculptured pocket object

points per square inch. Figure 6.7 shows the final reconstructed pool model after 14 iterations with sidewalls added. To verify the estimated error of the final model, 50 randomly selected points were measured. The maximum position error of these 50 points is 0.0270 inches and is less than the expected error.

6.2 Compressor Blade

The compressor blade is the two opposing blade surfaces on an aircraft engine compressor disk. It was designed and manufactured using the Alpha_1 modeler [3] and a three-axis machining center. This part includes several planar faces and one concave sculptured surface and presents a challange to the process because of the multiple and concave surfaces. The original measured object is shown in Figure 6.8.

Only the inside sculptured surface is reconstructed using the **SuRP**, Because other surfaces can be modeled using planar surfaces. The surface area of the inside surface is about 6.6 square inches. Initially, 12 points on three control curves were

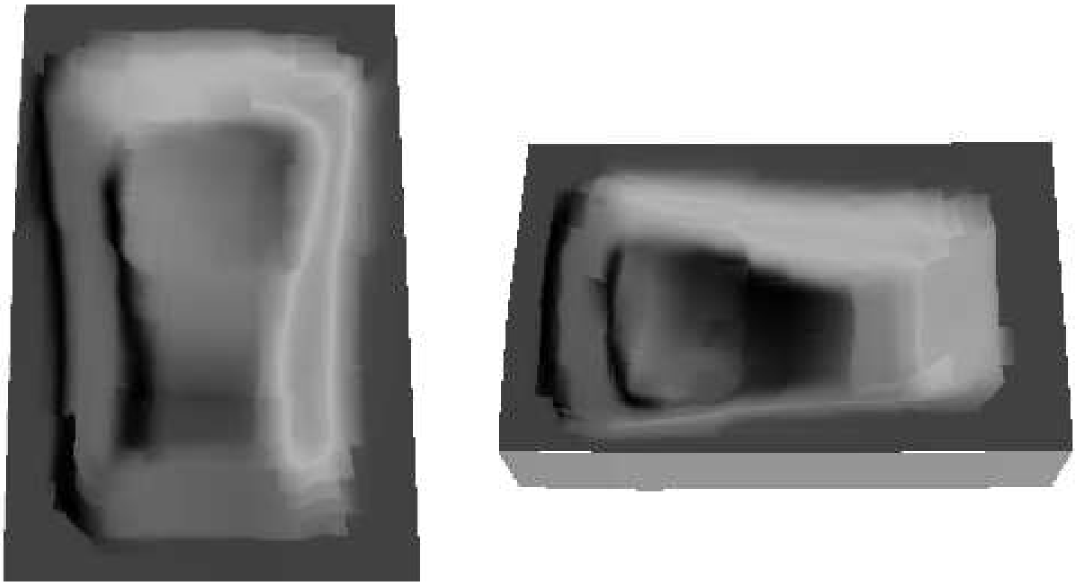

Figure 6.7. Two views of the reconstructed sculptured pocket model

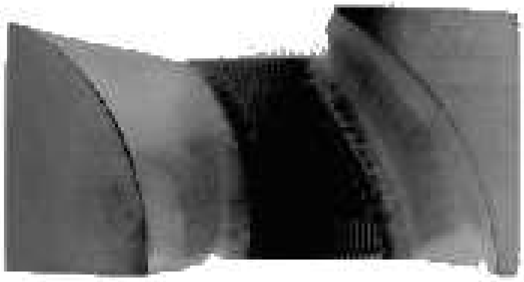

Figure 6.8. Original compressor blade object

used to produce the initial model. The linear control curves and the bilinear surface model are shown in Figure 6.9 and Figure 6.10. The initial model has maximum error of 0.5751 inches using cubic B-spline approximation.

Figure 6.11 shows the number of points at each iteration. Figure 6.12 shows the maximum, average and new measurement errors of the model at each iteration, computed using third order B-spline approximations. The efficiency measure of the internal obstacle avoidance algorithm is shown in Figure 6.13.

Approximately 2 hours were needed to perform 10 iterations of the reconstruction process. The model at the end of 10 iterations is shown in Figure 6.14. It has a model error of 0.0249 inches with 90% of points having error of 0.0118 inches or less. Fifty randomly sampled points returns maximum errors of 0.0166 inches.

6.3 Model of a Human Femur

The physical model of a distal end of the human femur was measured as one continuous surface. The original object is shown in Figure 6.15.

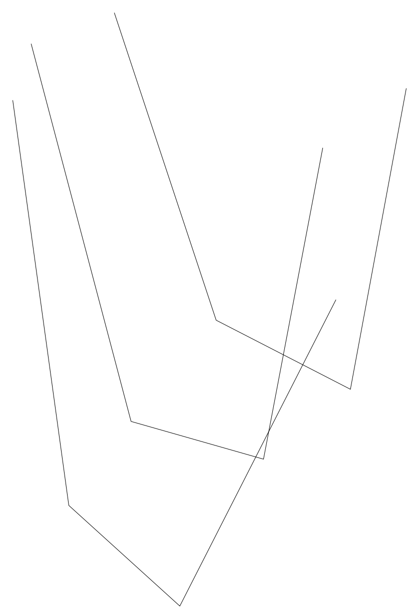

Figure 6.9. Initial control curves and control points for the compressor blade object

Figure 6.10. Initial compressor blade model

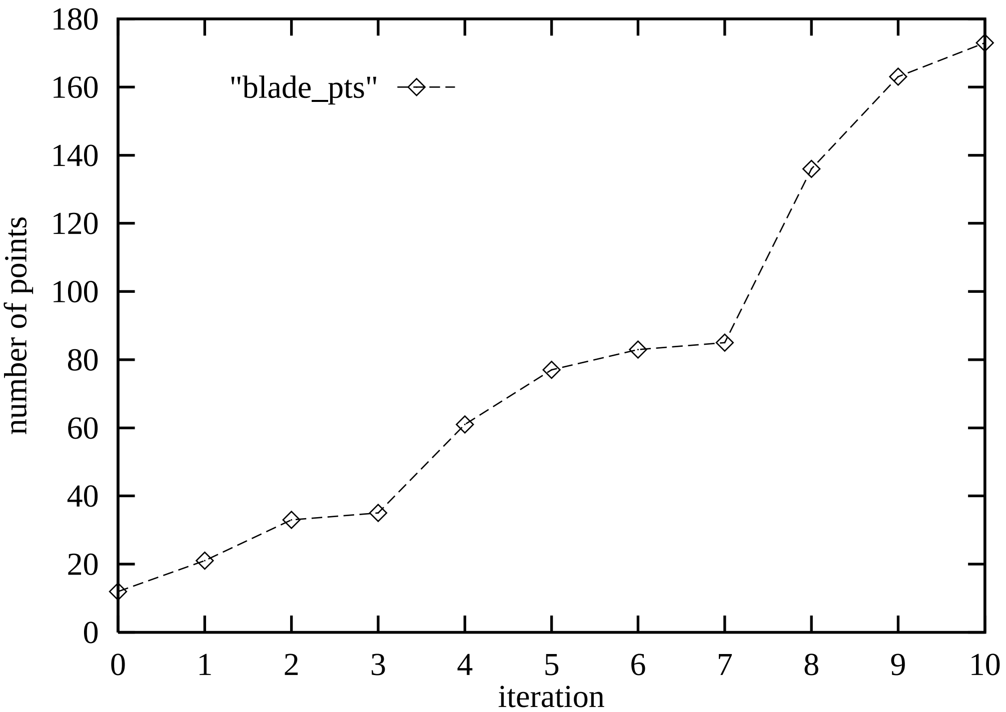

Figure 6.11. Plot of number of points at iteration for the compresssor blade object

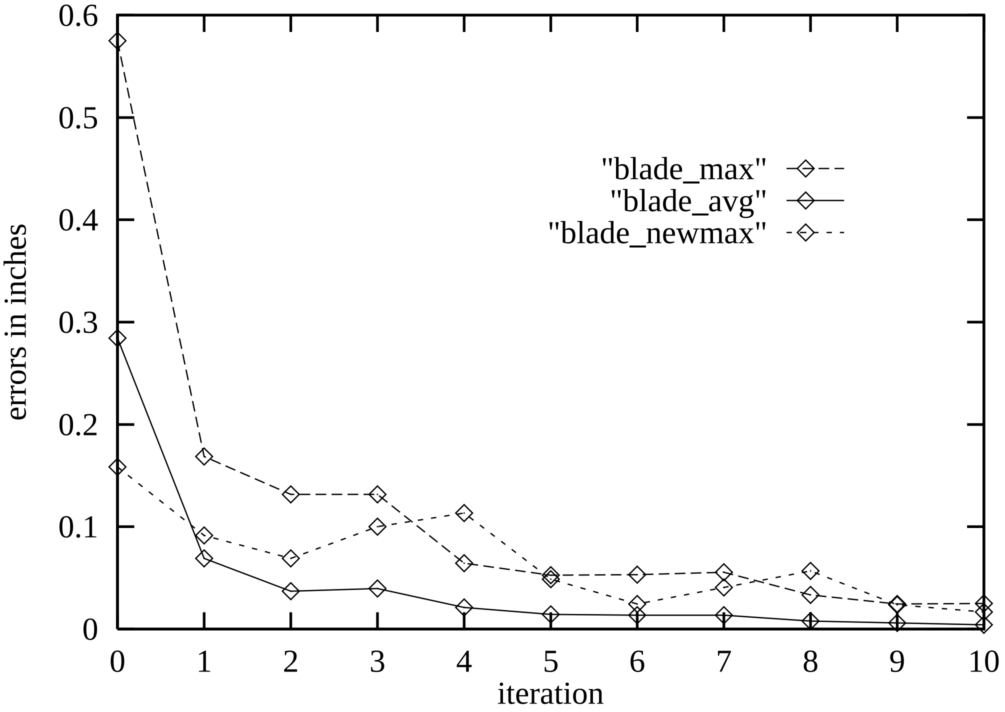

Figure 6.12. Plot of maximum and average error vs. iteration for the compressor blade object

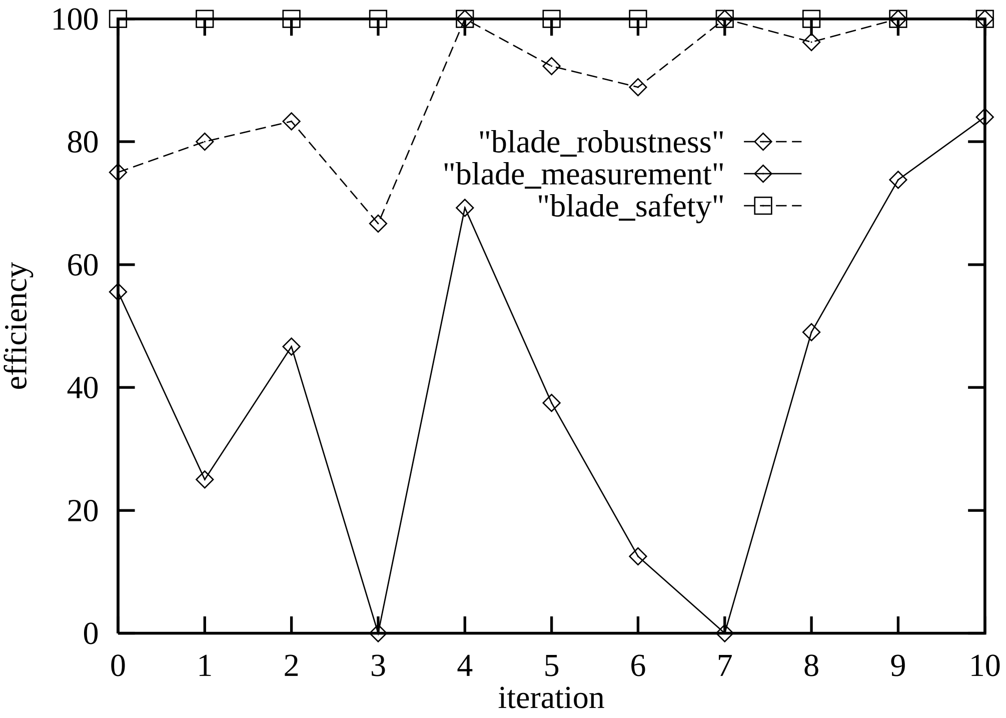

Figure 6.13. Plot of efficiency of the internal obstacle avoidance algorithm for the compressor blade object

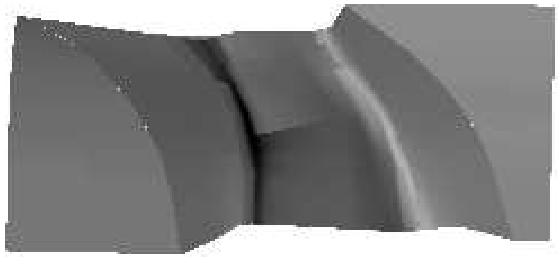

Figure 6.14. Reconstructed compressor blade model

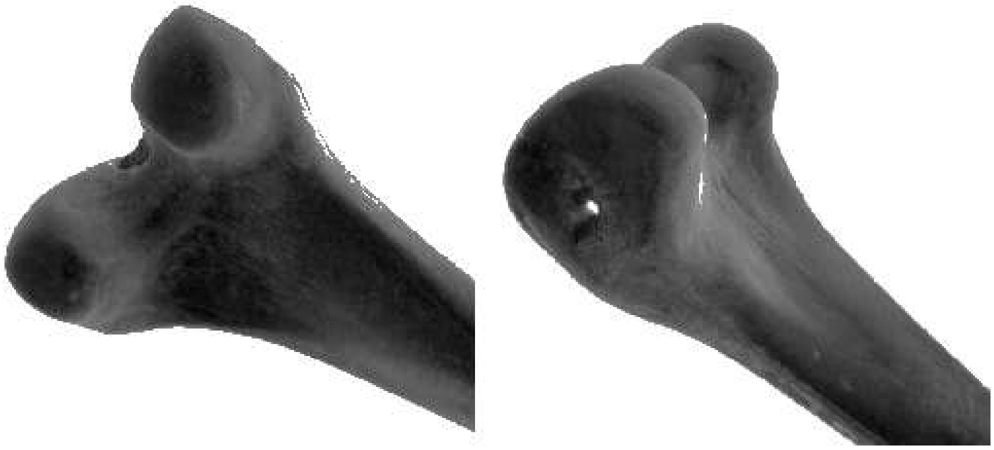

Figure 6.15. Original human femur object

The initial model included 56 points and maximum error of 0.5019 inches, and the initial model is shown in Figure 6.16.

Figure 6.17 shows the number of points at each iteration.

Figure 6.18 shows the maximum, average and new measurement errors of the model at each iteration, computed using third order B-spline approximations. The efficiency measure of the internal obstacle avoidance algorithm is shown in Figure 6.19.

The estimated surface area of the human femur object is about 35.1 square inches. The final model was reconstructed after 9 iterations with 520 points and maximum error of 0.0841 inches. The elapse time is about 7 hours. Fifty randomly sampled points at the end of 9 iterations shows maximum errors of 0.0784 inches. The reconstructed model is shown in Figure 6.20.

6.4 Model of a Human Vertebra

The physical model of the thoracic vertebral segment is shown in Figure 6.21.

The complexity of the vertebra makes it difficult to acquire the initial model. It is possible to model the object as one continous surface; however, with this approach,

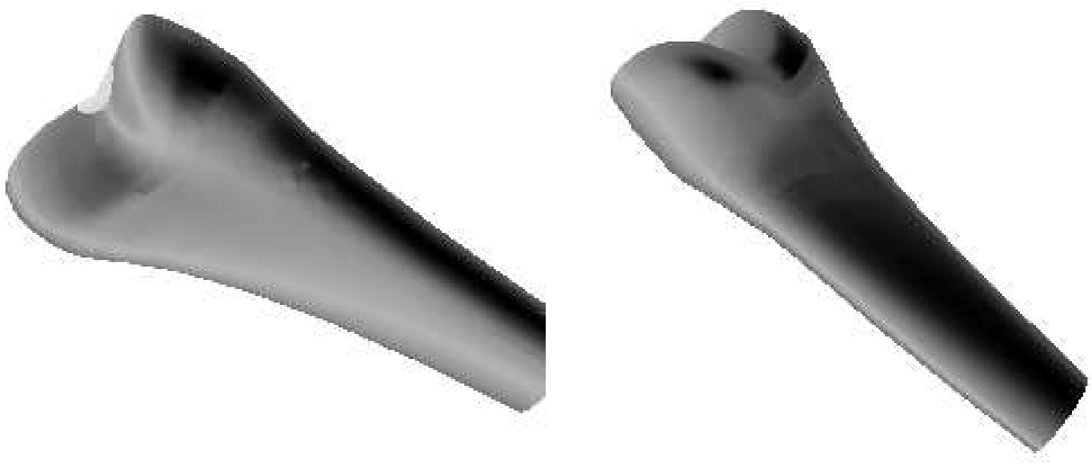

Figure 6.16. Initial human femur model

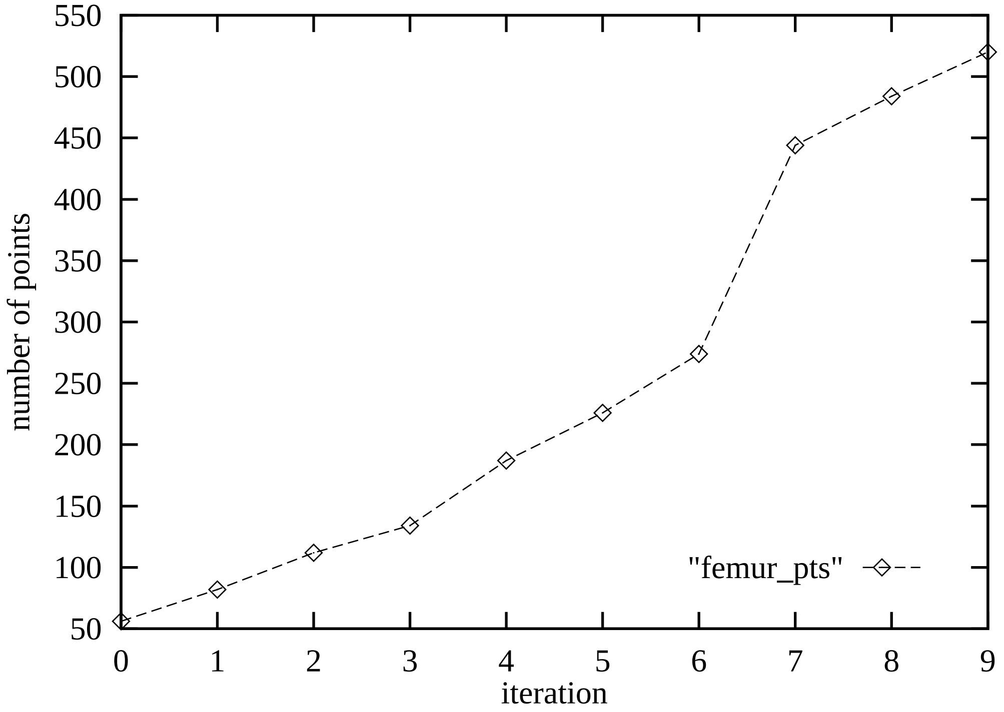

Figure 6.17. Plot of number of points at iteration for the human femur object

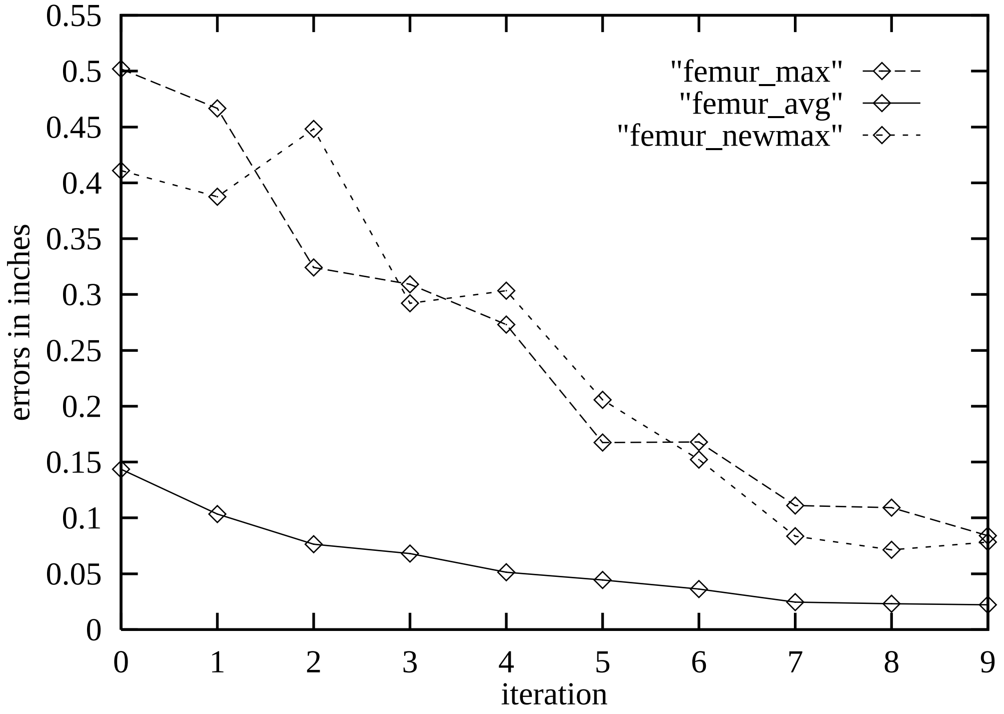

Figure 6.18. Plot of maximum and average error vs. iteration for the human femur object

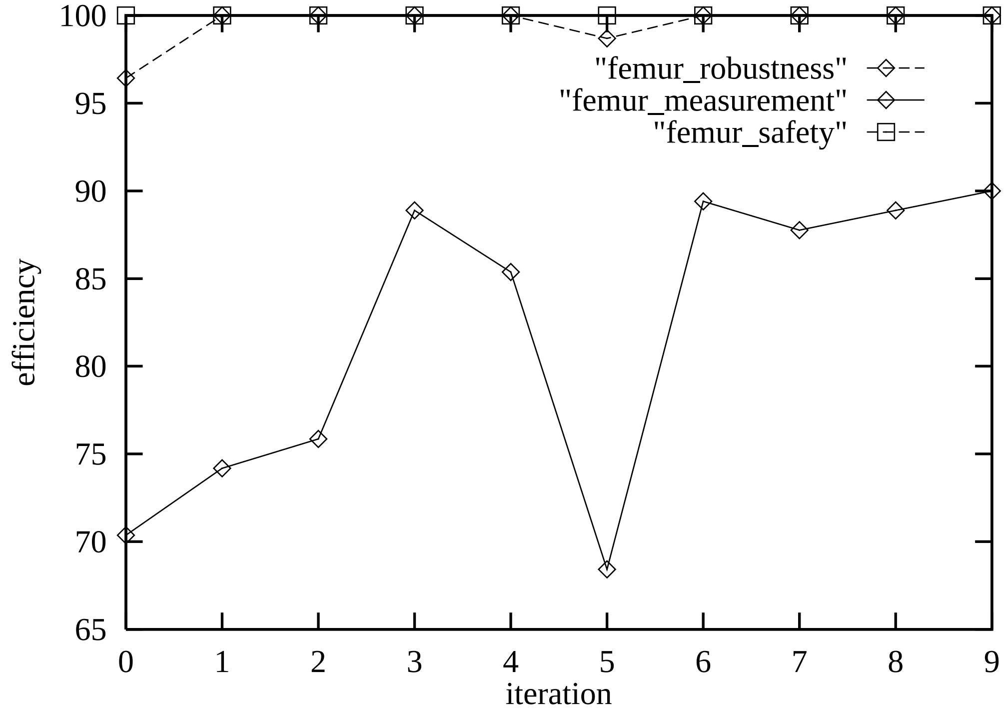

Figure 6.19. Plot of efficiency of the internal obstacle avoidance algorithm for the human femur object

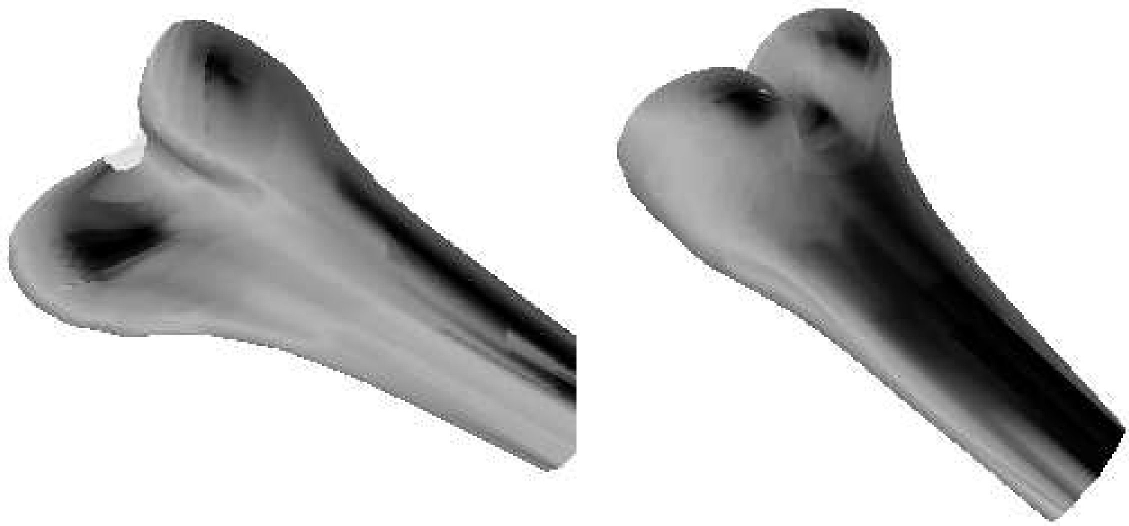

Figure 6.20. Final human femur model

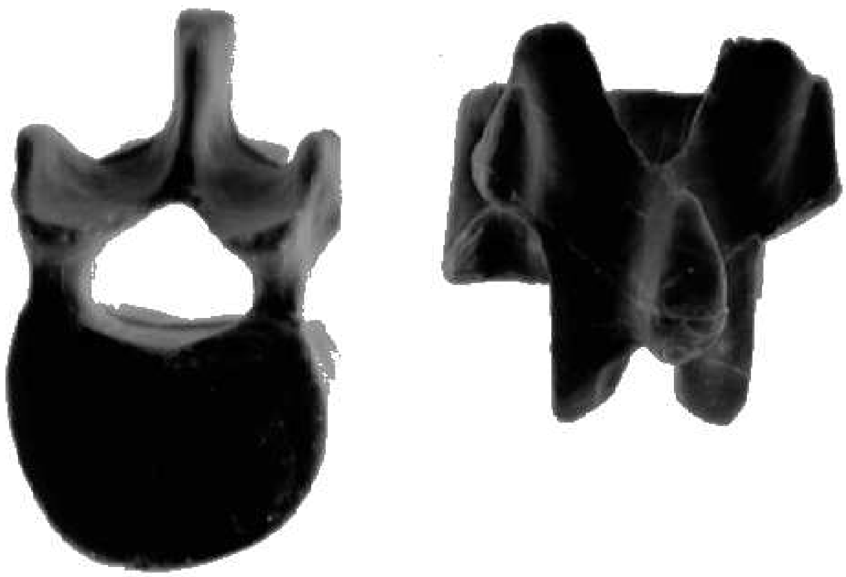

Figure 6.21. Original human vertebra object

a large number of points are required to achieve an usable initial model. On the other hand, if the vertebra object is decomposed into many smaller sections, the discontinuity between individal regions might not produce a visually pleasing model. Therefore, a compromise is reached, and the vertebra is measured in two sections: spine-T, which includes the transverse process, lamina and the spinous process of the segment, and spine-B, which includes the vertebral body. The estimated surface area of the vertebra object is about 14 square inches. The combination of high complexity and small physical dimension makes it a challenging case.

For spine-T, the initial model contains 143 data points, with maximum error of 0.2122 inches, and the initial spine-B model contains 102 points, with maximum error of 0.2939 inches. Figure 6.22 shows the initial solid model of the vertebra object.

Figure 6.23 shows the number of data points used to model the spine-T section at each iteration, and Figure 6.24 shows a plot of the maximum and average error at each iteration. In addition, the maximum error of new measurements using the

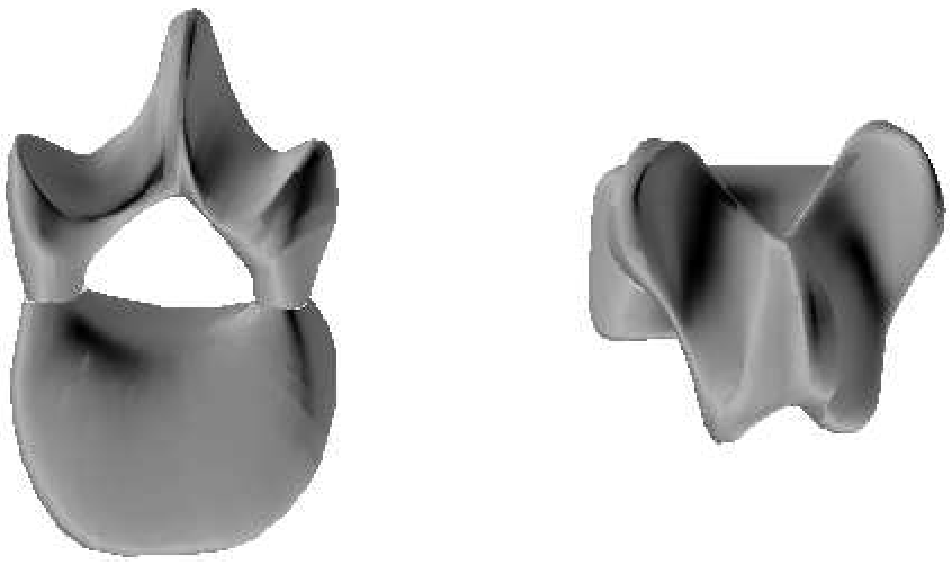

Figure 6.22. Two views of the initial human vertebra model

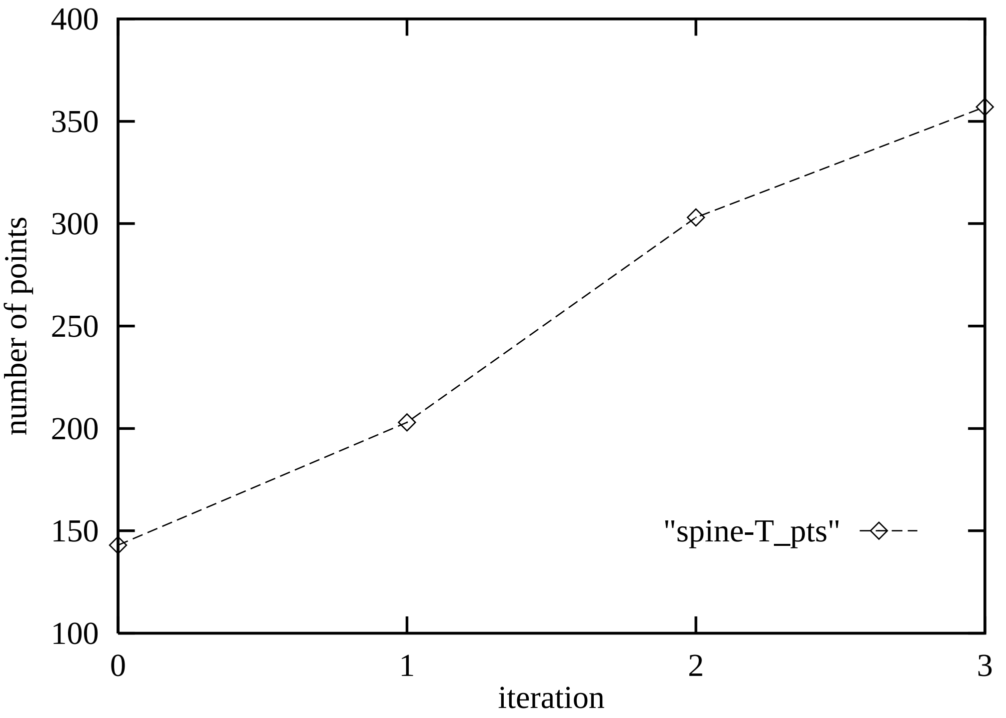

Figure 6.23. Plot of number of points at iteration for the spine-T section

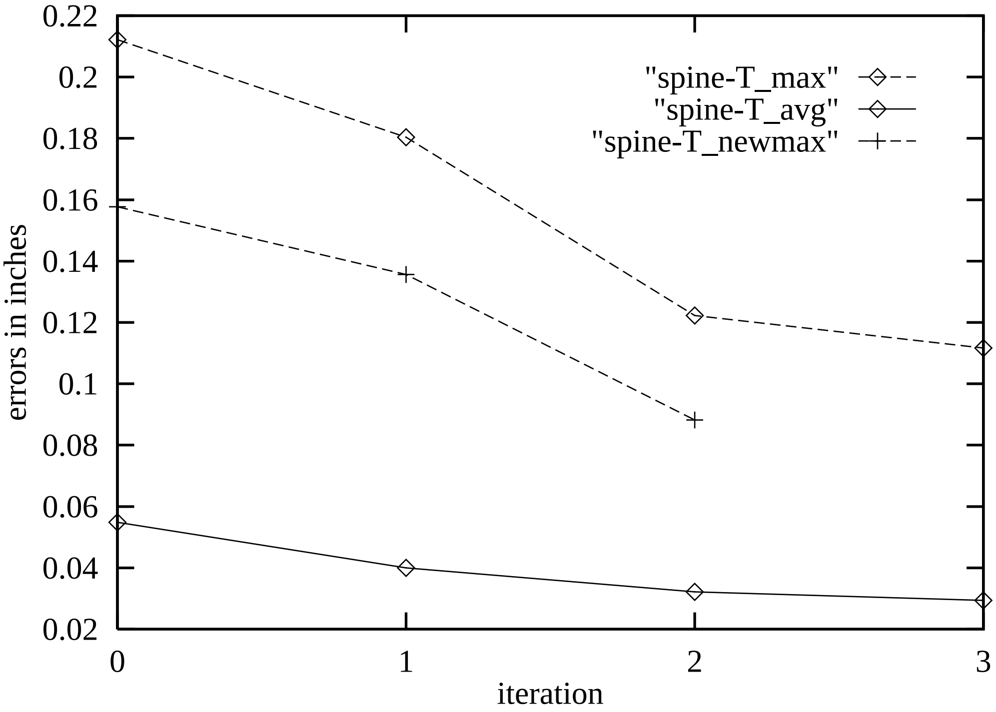

Figure 6.24. Plot of maximum and average error vs. iteration for the spine-T section

model at the iteration is also shown.

Figure 6.25 shows number of data points used to model the spine-B section at each iteration, and Figure 6.26 shows a plot of the maximum error, average error and new measurement errors at each iteration.

The efficiency measures of the internal obstacle avoidance algorithm is shown in Figure 6.27. The efficiency plot shows the combined results from both sections.

After approximately 10 hours and 3 iterations, the model error is reduced by about a half. The final Spine-T model contains 357 points with maximum error of 0.1117 inches, and spine-B section contains 295 points with maximum error of 0.0815 inches.

Figure 6.28 shows the reconstructed human vertebra model at the end of 3 iterations. Visually, it is difficult to determine significant improvements from the initial model. However, the final model is a better approximation by comparing the surface boundaries between the intial and final model.

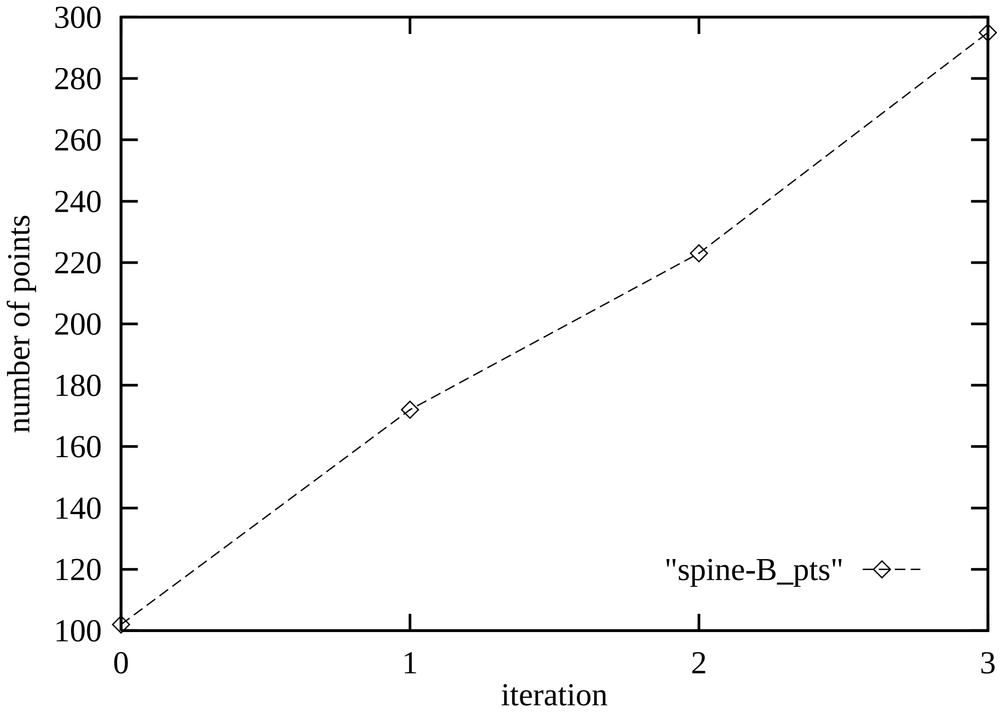

Figure 6.25. Plot of number of points at iteration for the spine-B section

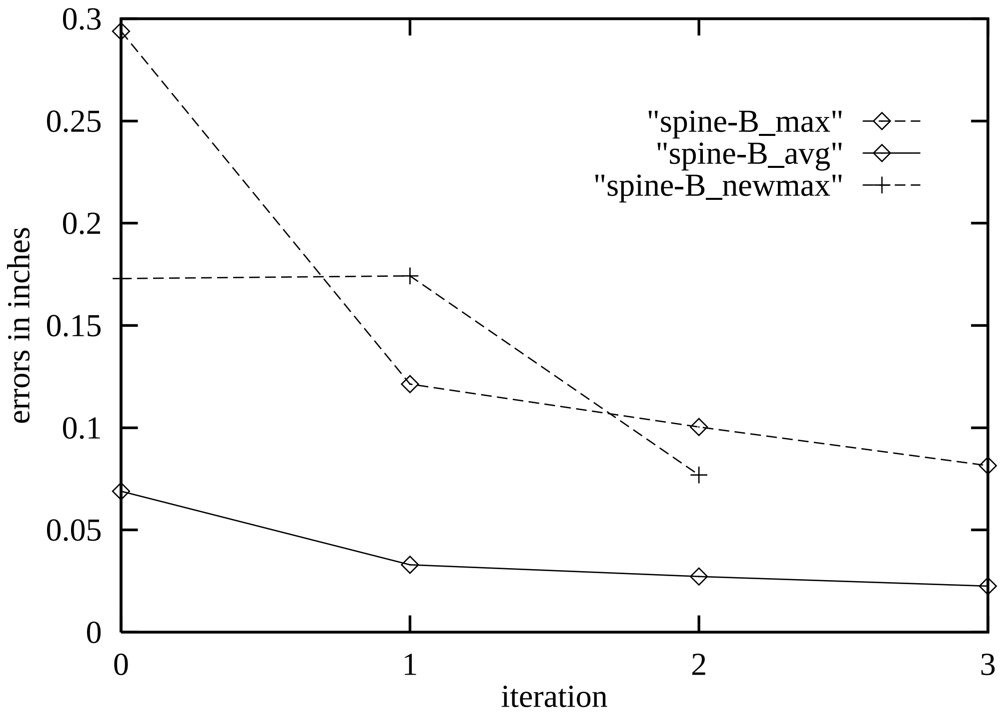

Figure 6.26. Plot of maximum and average error vs. iteration for the spine-B section

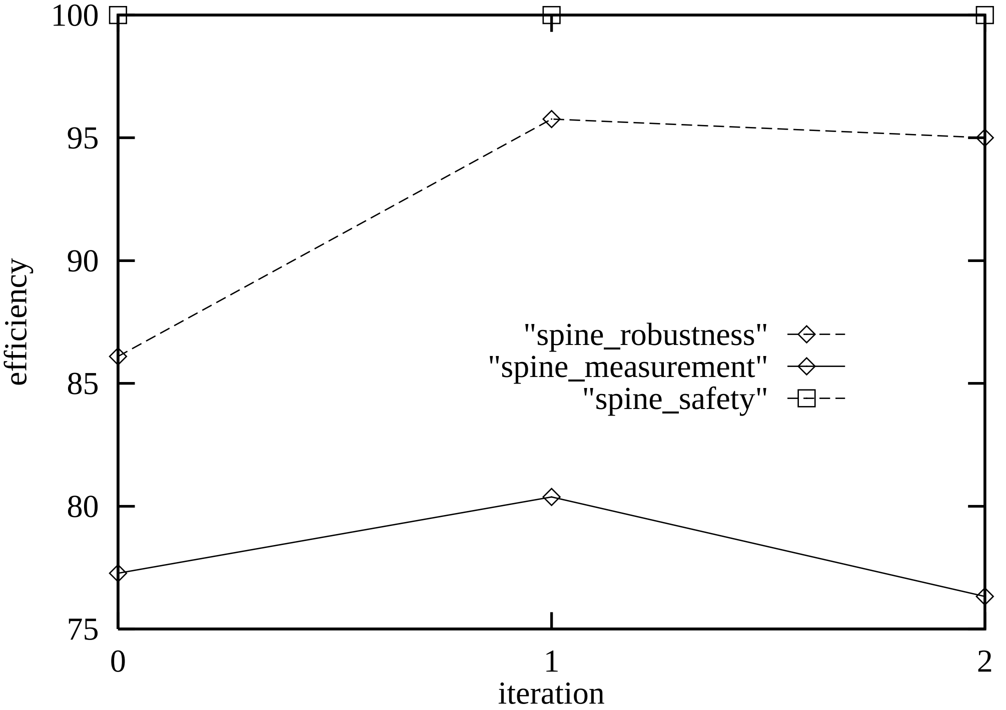

Figure 6.27. Plot of efficiency of the internal obstacle avoidance algorithm

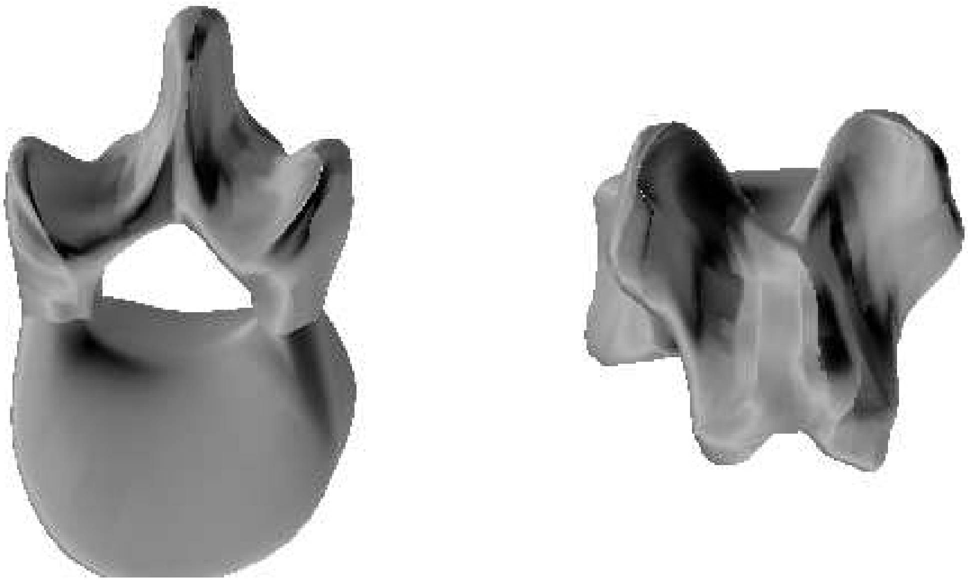

Figure 6.28. Two views of the reconstructed human vertebra model

6.5 Performance Evaluation

To validate the process, an evaluation of process performance is required. This evaluation is broken down into three components. First, the quality of the reconstructed model is discussed. Next, the elapsed time for reconstruction is presented. Finally, the efficiencies of the internal obstacle avoidance algorithm is investigated.

Table 6.1 shows the statistics of the reconstructed model. It presented the number of iterations (iter.) and number of points (npts) needed to reconstruct each object, the maximum (max err.), average (av er.) and standard deviation (SD) of *position errors* in inches for the final model, the error of 90% of the data points in inches, and the density of data points for each model in points per square inch. Except for one case, the maximum errors of reconstructed models are less than 0.1 inches. These values reinforced the visual quality of these models. These maximum errors represent the outlier error since 90% of all data points have error of about half of the maximum error. In all cases, the maximum errors are distributed around regions of large curvatures or the surface boundaries of the object.

Figures 6.5, 6.12 and 6.18 show the reduction of the maximum error, which approaches zero asymptotically as number of iteration increases. Figures 6.4, 6.11 and 6.17 show the number of data points increase exponentially with increasing

Table 6.1. Reconstructed model statistics

object	iter.	npts	max err.	av er.	SD	90%	density
pocket	14	587	0.0313	0.0069	0.0057	0.0149	40.96
blade	10	173	0.0249	0.0043	0.0043	0.0118	26.25
femur	9	520	0.0841	0.0222	0.0164	0.0477	14.82
spine-B	3	295	0.0815	0.0226	0.0159	0.0465	39.94
spine-T	3	357	0.1117	0.0294	0.0186	0.0542	54.11

number of iterations. These plots reflects the inherent limitation of the B-spline approximation technique. It is possible to decrease the maximum error of a model to reflect the capabilities of the CMM. However, with the current implementation, the number of data points required would be astronomical.

Table 6.2 shows the time, in minutes, needed to measured the parts to the final iteration, breaking down into three components: CPU measures the user time of the obstacle avoidance algorithm and the sampling plan, CMIS measures the time required to calibrate and probe sensors and physically perform measurements produced by the process, and STATS measures the time needed to perform the mappings and compute the statistics of the *model*.

The CMIS time shows that the time needed to measure one point stay relatively constant at approximately 0.13 minutes per measurement. This represents the fixed cost of the model reconstruction process using CMMs. Surprisingly, the mapping process accounts for the majority of the process time. This shows the inefficiency of the current implementation of the mapping algorithm and, perhaps, a disadvantage of using a B-spline approximation technique. The time required to perform internal obstacle avoidance varies according to the complexity of the objects being measured. Arguably, the most complex object being measured is the vertebra, and this is

Table 6.2. Total time required for each experiment

object	CPU (min)	CMIS (min)	STATS (min)	total (min)
pocket	58(12%)	173(37%)	242(51%)	473
blade	22(18%)	90(75%)	8(7%)	120
femur	23(6%)	135(33%)	248(61%)	406
vertebra	280(50%)	158(28%)	118(22%)	556
TOTAL	383(24.6%)	556(35.8%)	616(39.6%)	1555

reflected in Table 6.2 where CPU time accounts for 50% of the total time required.

The overall performance of the obstacle avoidance algorithm can be evaluated by combining results of all experiments. Table 6.3 shows the overall statistic of the measurements performed by the CMM. The table shows the number of points being considered for measurement (attempt), the number of valid measurements (meas), number of points that were not measured because the obstacle avoidance algorithm was not able to determine a safe probe orientation (not-mea), the number of measurement errors breaking down into three components as described in section 5.2.1, and the number of probe orientations used (ori). Table 6.4 shows the performance evaluation of the internal obstacle avoidance algorithm as defined by equation 3.4, 3.5 and 3.6 in section 3.1.

The internal obstacle avoidance algorithm has average robustness of 97.6%. However, from Figure 6.13 and Figure 6.6 shows that the majority of low robustness occurs at the early iterations, when the *model* has large *model error*.

On the average, CMIS executes one measurement every 0.13 minutes. There are a total of 2002 measurements performed, and 418 orientations were utilized. This implies that internal obstacle avoidance algorithm produces about one new probe orientation for approximately five measurements, or the algorithm has measurement efficiency of 79%. Because each calibration requires five measurements, the

Table 6.3. Measurement statistics

object	attempt	meas	not-mea	missed	collision	mismea	ori
pocket	738	734	4	0	0	0	122
blade	272	257	15	0	0	0	107
femur	565	563	2	0	0	0	92
vertebra	476	442	31	0	0	3	97
TOTAL	2051	1999	52	0	0	3	418

Table 6.4. Internal obstacle avoidance performance evaluation

object	robustness	measurement efficiency	safety
pocket	99.5%	83.4%	100%
blade	94.5%	58.4%	100%
femur	99.6%	83.7%	100%
vertebra	93.5%	78.2%	100%
TOTAL	97.6%	79.1%	100%

implemented internal avoidance algorithm reduces execution time by approximately 1 minutes for every 5 measurements performed. The measurement efficiency varies according to number of measurements attempted at one iteration, or it can be affected by the shape of the *object*. For example, Figure 6.6 shows the measurement efficiency at every iteration of reconstructing the sculptured pocket object. In iteration 1 and 4, the measurement efficiency is 0%; this means that every new measurement requires a different probe orientation. However, Figure 6.4 shows that a number of new measurement attempts at those iterations are small; three new measurements were taken at both iteration. The compressor blade object has the worst average measurement efficiency because the entire surface is concave. These four experiments show measurement efficiencies fluctuate between 0% and 94%.

The overall statistics of 98% robustness, 100% safety measure, and 79% measurement efficiency demonstrated the validity of obstacle avoidance algorithm and the hierarchial B-spline approximation, because these two issues are integrated in the reconstruction process.

CHAPTER 7

CONCLUSIONS

This paper presents an approach to capture data by moving a probe and avoiding collisions and to reconstruct sculptured surfaces using coordinate measuring machines. This approach uses the surface approximation and specification methods and refinement methodologies in the Alpha_1 system to model the data, reverse engineer multiple surfaces, and make a solid model from the modeled surfaces. An iterative approach is used to refine the B-spline surface model whose quality is evaluated using the position differences between the model and data points. The process iterates until all the position differences are below a user specified acceptable level. The current implementation has not been optimized. However, the integrated approach has already allowed issues in surface approximation, sampling plan, obstacle avoidances, and inspection to be investigated coherently and to produce a basis for comparison in further investigations.

The **SuRP** was tested using a variety of objects with different shape and sizes. Experimental results show that the system is an alternative to other object reconstruction systems. Overall, initial experiments have shown the following difficulties in the current version:

- The **SuRP** is time consuming to achieve an accurate model.
- The samples may not satisfy the Nyquist criteria. High frequency regions of the object may not be reconstructed.
- The system can run into the inherent limitation of the approximation technique in term of model accuracy.

- It is difficult to localize the surface boundaries, especially if the object has surface discontinuities.

- The surface mapping algorithm is inefficient.

However, the following advantages have been observered:

- Models produced using **SuRP** have better accuracy than current remote sensing methods.

- Every model produced using **SuRP** has an uncertainty measure. This uncertainty measure gives a tolerance value of the model.

- By utilizing an iterative approach, the accuracy of the model can be controlled. If a better model needs to be obtained, more iterations and data can be acquired to improve the model.

- Path planning and obstacle avoidance algorithms can be utilized for an inspection task.

- By utilizing full 5-axis capability, **SuRP** is able to measure some regions of the object that are occluded from view by using remote sensing methods. It can describe a variety of objects as one continuous surface.

- Using the control curves to approximate the data points allows distribution of the data points better than using the rectangular control mesh.

- Accepted models are randomly sampled to ensure accuracy.

- The system is modular. This means that each component can be improved independently.

- **SuRP** is intergrated with a powerful geometric modeler. This integration enables visualization and manipulation of data interactively to assure quality models.

CHAPTER 8

FUTURE DIRECTIONS

Because the quality and efficiency of the system depend on the solutions to the surface estimation, the surface representation issues need further work. A good surface representation reduces the number of points needed, affecting the sampling plan and evaluation criteria. It also affects the reliability of the obstacle avoidance routines. Currently, hierarchical approximation is used to account for a poor model. However, due to the inefficiency of the mapping process and the limitation of the approximation techniques, when the quantity of data points increases, there is a diminished return of model quality and exponential increase in the computation time. One possible solution is to use interpolating surface patches. When there is sufficient accuracy in the reconstructed model, the model can be decomposed into numerous interpolating surface patches. Each patch would encompass a region of the model with similar surface curvature, and the surface tangent of each interpolating patch can be controlled to ensure continuity between adjacent patches.

In data acquisition issues, efficiency is an obvious direction of further research; an optimal path search for the external path planning and a better search technique for internal obstacle avoidance should be investigated.

In addition to improving the efficiency of the various algorithms used in this system, another direction of further research and development is in the area of sensors fusion. By using a remote sensor, in conjunction with the CMM, it is hoped that the overall system would offer the following advantages:

- Remote sensors can provide the initial estimate of the object and the environment.

- Accuracy of the model provided by remote sensors can be improved using the CMM. Accuracy of the model provided by the CMM can also be improved by remote sensors.

- Remote sensors can monitor the progress of the CMM manipulation and provide a warning if collision is imminient.

APPENDIX

SAMPLE CMIS PROGRAM

```
$$
$$ CMIS program initializations
$$
dmismn/'p_p_t1.0 3.22'
filnam/'on'
FILNAM/COORD,'setcal.crd',INPUT
wkplan/xyplan
prcomp/on
finpos/off
units/inch,angdec
mode/auto,prog,man
fedrat/posvel,mpm,9.2
$$
$$ Move probe to a default home position.
$$
GOTO/15,15,20
filnam/sens,'p_t1_0.sns',output,OVERWR
filnam/sens,'p_t1_0.sns',input
$$
$$ Setting physical speed of the probe
$$
fedrat/posvel,mpm,4.8
fedrat/mesvel,mpm,0.65
$$
$$ Setting approach, retract, and search distances.
$$
SNSET/apprch,0.5
SNSET/retrct, 0.5
SNSET/search, 0.5
SAVE/D(MCS)
S(HOME)=SNSDEF/PROBE,INDEX,POL,90.0,0.0,0,0,-1,5.839000,0.078500
SAVE/S(HOME)
$$
$$ Calibrating a probe orientation
$$ with pitch angle of 45 degrees and
$$ roll angle of 105 degrees using a calibration ball.
```

```
$$
S(45P105)=SNSDEF/PROBE,INDEX,POL,45.000000,105.000000,0,0,-1,
  $5.839000,0.078500
SNSLCT/S(45P105)
RECALL/D(CAL)
F(CALBALL)=FEAT/SPHERE,OUTER,CART,0,0,0,1
CALIB/SENS,S(45P105),F(CALBALL),5
ENDMES
SAVE/S(45P105)
RECALL/D(MCS)
RECALL/S(HOME)
SNSLCT/S(HOME)
GOTO/15, 15, 20
$$
$$ more probe can be calibrated
$$
$$ Measure a point at [12.97430,19.30422,1.27942]
$$ with an approach vector of [0.84184,-0.12044,0.52373].
$$
RECALL/S(45P105)
SNSLCT/S(45P105)
$$
$$ Moving around various obstacles
$$ computed by the external path planning routine.
$$
GOTO/11.011889,8.092389,15.871204
GOTO/11.011889,8.092389,15.871204
GOTO/12.339640,14.648429,2.800479
GOTO/12.339640,18.018429,2.800479
GOTO/13.849175,19.168660,1.823704
$$
$$ Setting approach, retract, and search distances.
$$
SNSET/APPRCH,1.000000
SNSET/RETRCT,1.000000
SNSET/SEARCH,1.000000
F(P10_15)=FEAT/POINT,CART,12.97430,19.30422,1.27942,$
  0.84184,-0.13044,0.52373
MEAS/POINT,F(P10_15),1
PTMEAS/CART,12.97430,19.30422,1.27942,0.84184,-0.13044,0.52373
ENDMES
OUTPUT/FA(P10_15)
$$
$$ More points can be measured.
```

```
$$
ENDFIL
```

REFERENCES

[1] P. J. Besl and R. Jain, "Three-dimensional object recognition," *ACM Computing Surveys*, vol. 17, pp. 75–145, Mar. 1985.

[2] R. M. Bolle and B. C. Vemuri, "On three-dimensional surface reconstruction methods," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 1–13, Jan. 1991.

[3] University of Utah, *Alpha_1 User's Manual*, 1992.

[4] T. Tsujimura, T. Yabuta, and T. Morimitsu, "Shape-reconstruction system for three-dimensional objects using an ultrasonic distance sensor mounted on a manipulator," *J. Dynam. Syst., Measurement, Control*, vol. 111, pp. 180–186, June 1989.

[5] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging.* New York: IEEE Press, 1988.

[6] J.-L. Saint-Marc, P. Jezouin, and G. Medioni, "A versatile pc-based range finding system," *IEEE Trans. Robotics Automat.*, vol. 7, pp. 150–256, Apr. 1991.

[7] T. Moring, I. Heikkinen, and R. Myllyla, "Acquisition of three-dimensional image data by a scanning laser rang finder," *Optical Engineering*, vol. 28, pp. 897–902, Aug. 1989.

[8] S. Watanabe and Y. Masahide, "An ultrasonic visual sensor for three-dimensional object recognition using neural networks," *IEEE Trans. Robotics Automat.*, vol. 8, pp. 36–55, Apr. 1992.

[9] O. D. Faugeras and M. Hebert, "The representation, recognition, and locating of 3d objects," *The Int. J. Robotics Research*, vol. 5, no. 3, pp. 27–52, 1986.

[10] P. K. Allen, "Sensing and describing 3-d structure," in *Proc. IEEE Int. Conf. Robotics Automat.*, pp. 126–131, IEEE, Apr. 1986.

[11] R. E. Ellis, "Acquiring tactile data for the recognition of planar objects," in *Proc. IEEE Int. Conf. Robotics Automat.*, pp. 1799–1805, IEEE Press, Apr. 1987.

[12] P. C. Gaston and T. Lozano-Perez, "Tactile recognition and localization using object models: The case of polyhedra on a plane," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, no. 3, pp. 257–265, 1984.

[13] H.-T. Menq, C.-H. Yau, and G.-Y. Lai, "Automated precision measurement of surface profile in cad-directed inspection," *IEEE Trans. Robotics Automat.*, vol. 8, pp. 268–278, Apr. 1992.

[14] K. C. Sahoo and C.-H. Menq, "Localization of 3-d objects having complex sculptured surfaces using tactile sensing and surface description," *J. Engineering for Industry*, vol. 113, pp. 85–92, Feb. 1991.

[15] Y. L. Xiong, "Computer aided measurement of profile error of complex surfaces and curves: theory and algorithm," *Int. J. Mach. Tools Manufact.*, vol. 30, no. 3, pp. 339–357, 1990.

[16] F. L. Merat and G. M. Radack, "Automatic inspection planning within a feature-based cad system," *Robotics and Computer Integrated Manufacturing*, vol. 9, no. 1, pp. 61–69, 1992.

[17] D.-P. Lee, A.-C. Chen, and C.-L. Lin, "A cad/cam system from 3d coordinate measuring data," *Int. J. Production Research*, vol. 28, no. 12, pp. 2353–2371, 1990.

[18] W.-L. Kwok and P. J. Eagle, "Reverse engineering: extracting cad data from existing parts," *Mechanical Engineering*, pp. 52–55, Mar. 1991.

[19] Fanamation, *Comero Training Manual*, 1990.

[20] R. A. Brooks, "Planning collision-free motions for pick-and-place operations," *Int. J. Robotics Research*, vol. 2, pp. 19–44, 1983.

[21] T. M. Rao and R. C. Arkin, "3d navigational path planning," *Robotica*, vol. 8, pp. 195–206, Jul.–Sep. 1990.

[22] D. Zhu and J.-C. Latombe, "New heuristic algorithms for efficient hierarchial path planning," *IEEE Trans. Robotics Automat.*, vol. 7, pp. 9–20, Feb. 1991.

[23] S. Jun and K. G. Shin, "Shortest path planning in discretized workspaces using dominance relation," *IEEE Trans. Robotics Automat.*, vol. 7, pp. 342–350, June 1991.

[24] H. Hirukawa and S. Kitamura, "A collision avoidance method for robot manipulators based on the safty first algorithm and the potential function," *Advanced Robotics*, vol. 4, no. 1, pp. 43–57, 1990.

[25] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Trans. Robotics Automat.*, vol. 8, pp. 23–32, Feb. 1992.

[26] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Trans. Robotics Automat.*, vol. 8, pp. 338–349, June 1992.

[27] Fanamation, *Measurement Inspection Software (CMIS) Command Reference Manual*, 1990.

[28] D. Hearn and M. P. Baker, *Computer Graphics*, section 10.6, pp. 215–217. Englewood Cliffs, New Jersey: Prentice Halls, Inc., 1986.

[29] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, New Jersey: Prentice Halls, Inc., 1982.

[30] D. Terzopoulos, "Multilevel computational processes for visible-surface reconstruction," *Computer Vision, Graphics and Image Processing*, vol. 24, pp. 52–96, 1983.

[31] S. S. Sarvajit and B. G. Schunck, "A two-stage algorithm for discontinuity-preserving surface reconstruction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 36–55, 1992.

[32] L. L. Shumaker, "Fitting surfaces to scattered data," in *Approximation Theory II* (C. K. Lorentz, G. G. Chui and L. L. Schumaker, eds.), pp. 203–267, New York: Academic Press, 1989.

[33] D. Hearn and M. P. Baker, *Computer Graphics*, chapter 10, pp. 189–216. Englewood Cliffs, New Jersey: Prentice Halls, Inc., 1986.

[34] R. E. Barnhill and R. F. Riesenfeld, eds., *Computer Aided Geometric Design*. New York: Academic Press, 1974.

[35] I. Elshennawy, A. K. Ham, and P. H. Cohen, "Evaluating the performance of coordinate measuring machines," *Quality Progress*, pp. 59–65, Jan. 1988.

[36] T. Arun, K.S. Huang, and S. Blostein, "Least-squares fitting of two 3d point sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, pp. 698–700, Sept. 1987.

[37] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 239–256, Feb. 1992.

[38] A. Oppenheim and R. Schafer, *Discrete-time signal processing*, chapter 3, pp. 80–130. Prentice hall sgnal processing series, Englewood Cliffs, New Jersey: Prentice Hall, 1989.

[39] M. Marvasti, F. Analoui, and M. Gamshadzahi, "Recovery of signals from nonuniform samples using iterative methods," *IEEE Trans. Signal Processing*, vol. 39, pp. 872–878, Apr. 1991.