Abstract

BOONMA, APICHART. Haptic-Based Sharp Edge Retaining and Gap Bridging Algorithms for Computer Aided Design (CAD) and Reverse Engineering (RE). (Under the direction of Dr. Yuan-Shin Lee).

The objective of this paper is to develop a computational and haptic interface technique to improve the repair of non-watertight virtual models through the enhancement of hole filling, sharp edge retaining and gap bridging strategies. In Reverse Engineering (RE), a virtual model is created from point cloud data using surface fitting or triangulation techniques. Point cloud or three dimensional position data is usually acquired by laser scanning a physical object. Obtaining data that well characterizes the part surface is quite difficult. Defective point cloud that is caused by loss or non-uniform distribution of scanning points might lessen the quality of the resulting virtual model and these defects usually appear as holes or gaps on the constructed triangulate facet surfaces. Such model is usually called a non-watertight model.

Watertight triangulate facet model is very important for many applications, such as reverse engineering (RE), computer-aided design (CAD), rapid prototyping (RP) and NC machining. How to fix and repair the non-watertight triangulate facet models with defects is vital for computer aided design (CAD) and reverse engineering (RE).

This thesis focuses primarily on developing a technique to repair holes and gaps on the triangulate facet models. Haptic-based sharp edge retaining and gap bridging algorithms are presented in this paper. Our haptic-based hole filling algorithm can be used to selectively repair only the defective holes chosen by the user. The proposed haptic-based sharp edge retaining algorithm is able to retain sharp edge for both geometrical and free-formed models.

To seal the gap between two surfaces, a haptic-based gap bridging algorithm is proposed.

The proposed algorithms were implemented with a 6-DOF (degree of freedom) PHANTOM Desktop™ haptic device at our research lab. The haptic interface device is used to provide force and torque feed back to the user and to provide accessibility to the surface of the input model. The user can feel the surface of the model, select the hole that needs to be repaired, specify parameters in order to construct sharp edge, and bridge the gap that appears in the surface model. All the works are performed in real time basis. The implementation results show that our techniques can be used to repair holes and gaps in a non-watertight model and transform the model into a watertight one. The proposed techniques can be used in computer aided design (CAD) and reverse engineering (RE) applications.

Haptic-Based Sharp Edge Retaining and Gap Bridging Algorithms for Computer Aided Design (CAD) and Reverse Engineering (RE)

By

Apichart Boonma

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Industrial Engineering

Raleigh
2006

Approved By:

Dr. Yuan-Shin Lee
Chair of Advisory Committee

Dr. Ezat T. Sanii

Dr. Stephen D. Roberts

To my parents, Prasit and Thongsri

Biography

Apichart Boonma was born in Khon Kaen, Thailand. He is a M.S. student in the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University, U.S.A. He received his B.S. (1999) degree in Industrial Engineering from Khon Kaen University, Thailand. His research interests include reverse engineering (RE), computer aided design and manufacturing (CAD/CAM), computational geometry for design and manufacturing and human-haptic interface. Upon the completion of his M.S. degree, he plans to continue his Ph.D. graduate study at the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University.

Acknowledgements

I would like to express my deep appreciation and gratitude to my advisor, Dr. Yuan-Shin Lee for his patience, suggestion and encouragement during my academic and research work at North Carolina State University. Without his guidance and continuous support, I would never have finished my study. I would like to thank Dr. Ezat T. Sanii and Dr. Stephen D. Roberts for their support and serving in my Master Thesis Committee.

I highly appreciate and would like to thank my research group members: Dr. Yongfu Ren, Dr. Susana K. Laiyuen, Shiyong Lin, Deyao Ren, Yingjie Li, Ronald L. Aman, Plawut Wongwiwat and Eui Seok Kim for their valuable discussions and suggestions.

I would also like to thank my friends: Pueng, Lena, Ann, Kaew, Pom and May for their discussions, support and encouragement.

Finally, I greatly appreciate my granddad, dad, mom, sister and my nieces for their love, encouragement and support.

Table of Contents

List of Figures

Page

Chapter 1

Introduction

1.1 Motivation

"If we can create digital duplicate of our world as easily as we can take a picture of what we see, the biggest breakthrough of the 21st century will be in manufacturing"

[Ping Fu, president and CEO of Raindrop Geomagic]

Reverse engineering (RE) is an approach consisting of series of processes, e.g. data capturing, preprocessing, surface fitting and CAD model generation, intended to create a virtual object model of a physical part. It has been widely used in several areas such as design of product where the drawing and CAD data of the product are unavailable, inspection and quality control (to compare the CAD model with its physical object), production of CAD data for highly complicated part, observation and investigation of accident or crime scene, and etc. In the past few decades, various researches have been done in order to effectively and efficiently generate the virtual model that best approximates its prototype out of a set of data. This data, so called scattered, unorganized point or point cloud, describes surface of an object in the form of x, y and z coordinates and is usually obtained by laser scanning the physical part. An example of a point cloud can be found in Figure 1.1.

In computer aided design (CAD) applications, it is strongly recommended to have a virtual model that is watertight. A watertight model is basically an object model represented by seamless surfaces (no defective holes or gaps). It is required as an input for CAD/CAM/CAE or other design and manufacturing software packages.

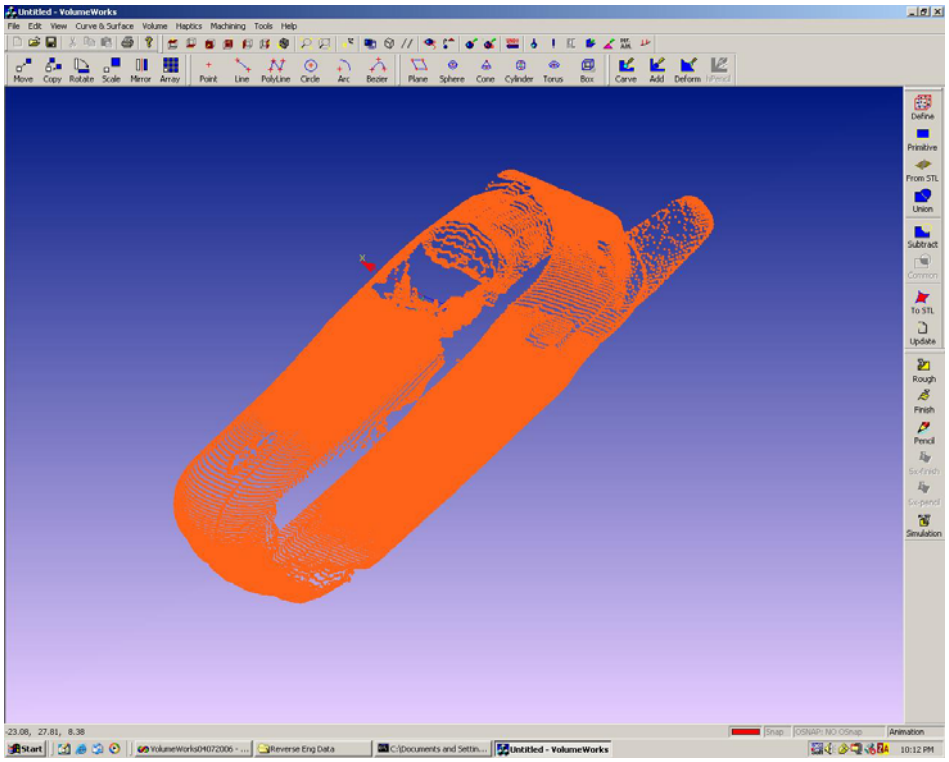


Figure 1.1 Point cloud data of a mobile phone.

In reverse engineering (RE), although data acquisition is critical, the biggest challenge is the segmentation and surface fitting, which is the process of generating a surface that best describes a set of scanned point data [Varady 1997]. Surface fitting is also called object reconstruction due to its function to re-create surfaces of an object. Why is object reconstruction so important? It is because the process itself is usually difficult to achieve since there are several factors involved. These factors might cause surface fitting process to render incorrect surfaces, which most likely appear as defective holes and gaps as shown in Figure 1.2. The factors causing incorrect rendering might result from the laser scanning process, shiny surface of the physical parts, parts complexity (undercuts), and etc. Therefore, it is required that surface fitting algorithm must be effective, efficient and robust enough to overcome these factors, such that the final outcome is a watertight model that best defines the shape of its prototype. However, such powerful surface fitting

algorithm does exist today. To get around the problems, researchers proposed post processing techniques that are able to repair and turn a non-watertight model into the watertight one. These processes may consist of hole filling, triangle subdivision, surface smoothing and etc.

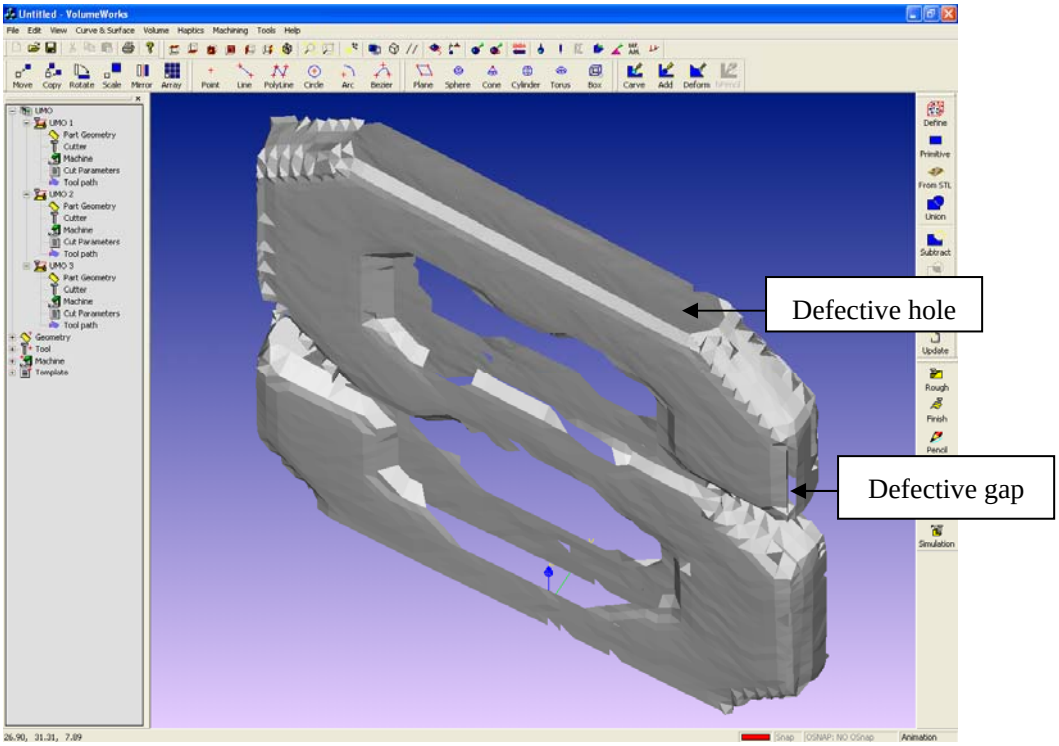


Figure 1.2 An example model with defects

In hole repair process, holes appearing in a model could be classified into 2 major categories; (1) functional holes, as shown in Figure 1.3, that are required to remain in the model to perform some design functions, (2) defective holes that must be repaired. Hole filling algorithm should be able to selectively repair only the defective holes. This requirement is difficult to achieve automatically since the algorithm may not be able to differentiate a defective hole from the functional one. Therefore, human interaction

should be applied to help selecting a target hole, then the hole filling algorithm repair the hole in automatic fashion.

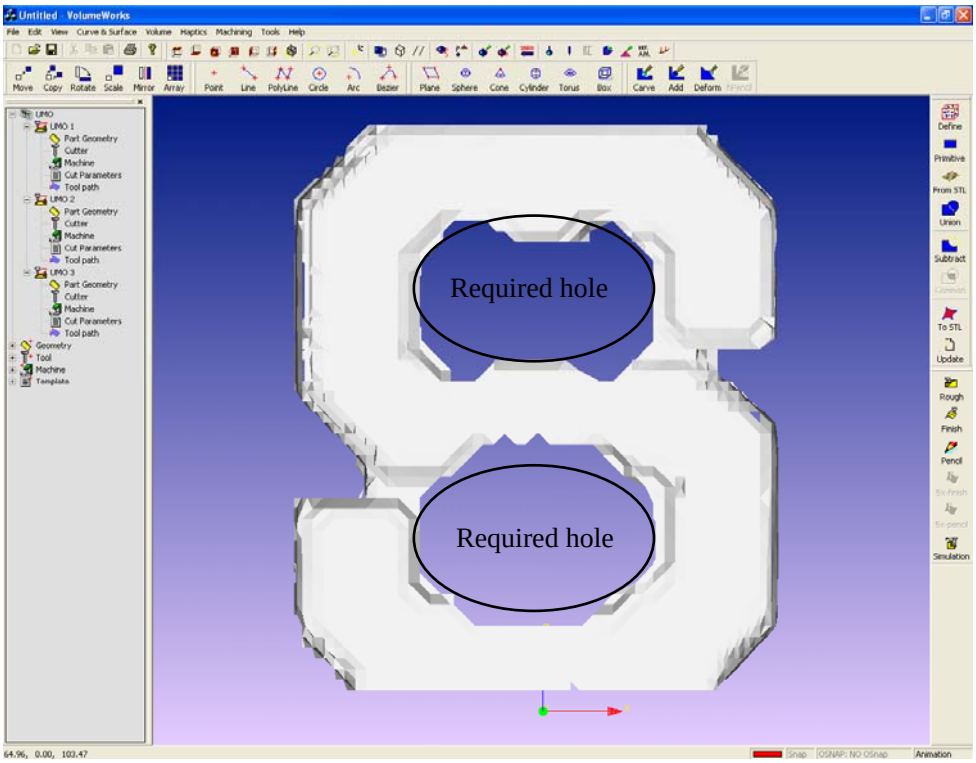


Figure 1.3 An example model with some holes left intentionally.

Moreover, if a repaired surface (the surface that is created in order to fit and eliminate a selected hole) is required to represent sharp edge, the repairing algorithms should be able to extract information from hole boundary and create such feature onto the repaired surface. Some researches presented sharp edge retaining algorithm that could be only applied to geometric parts (e.g. industrial parts), where two or more considered surfaces are in presence with large difference in their normal vectors. An example of such parts is

shown in Figure 1.4. To the best of our knowledge, there is no existing algorithm that is able to handle the cases where the normal vectors of the surfaces at sharp edge are not much different, such as the free-form object shown in Fig 1.5.

In this paper, we propose haptic-based hole filling and sharp edge retaining algorithm to repair the defective holes and generate sharp edges on complex surface models.

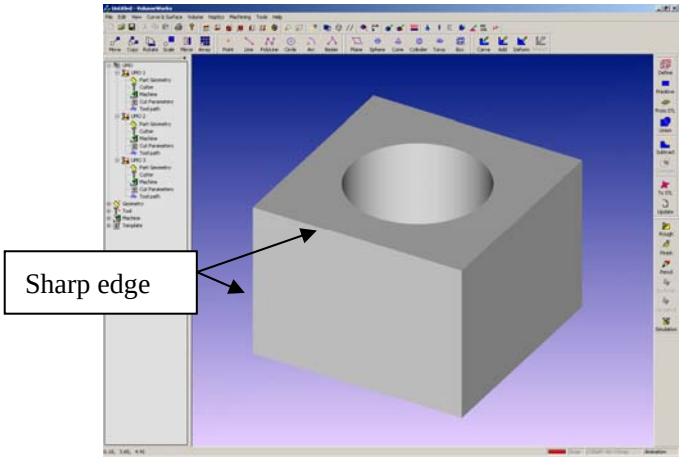


Figure 1.4 A geometrical part whose triangular patches near sharp edge have very different surface normal vectors.

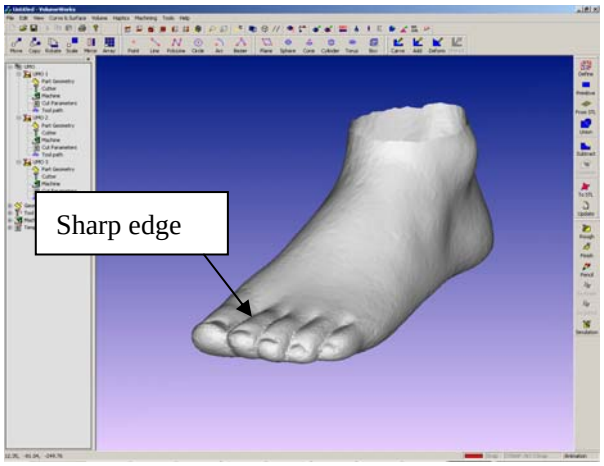


Figure 1.5 Unlike an industrial part, a free form model consists of several triangular patches that does not have much different surface normal vector.

1.2 Research Objectives

The objective of this paper is to investigate and develop computational and haptic interface techniques to retain sharp edges, fill holes and bridge gaps in incomplete models. Our study aims to provide a tool for holes and gaps repair and sharp edge retaining using computational geometry and haptic interface. Some major research tasks are focused in this paper:

- Haptic-based Hole Filling Algorithm. This haptic-based algorithm provides choices for users to selectively repair defective holes in incomplete surface model. The algorithm, except at the stage of selecting to-be-filled hole, works automatically.
- Haptic-based Sharp Edge Retaining Algorithm. This haptic-based algorithm provides a tool for creating sharp edge over reconstructed meshes in order to retain original shape of the model. It is performed in semi-automatic fashion.
- Haptic-based Gap Bridging Algorithm. This approach uses haptic interface device to seal gaps between two surfaces. It can be classified into two sub-methods consisting of manual gap bridging and semi-automatic gap bridging. Manual gap bridging is a method that users control haptic interface device to gradually stitch the gap, while semi-automatic one is integration between the manual method and hole filling algorithm.

Haptic-based hole filling approach has some advantages over the conventional hole filling algorithm in that, it allows users to selectively fill only the defective holes. The proposed haptic-based sharp edge retaining algorithm helps creating sharp edge based on boundary features and user-defined parameters, and can be used with simple hole filling algorithm. It is also more flexible and intuitive compared to the previous methods since users can see and create sharp edge by themselves. We also introduce herewith the haptic-based gap bridging algorithm that could be used to stitch two or more surfaces together. The approaches can be applied to reverse engineering, product design, virtual model repair and other computer aided design (CAD) applications.

1.3 Thesis Organization

The remainders of this thesis are organized as follows:

Chapter 2 presents a literature review on related research issues including object reconstruction, hole filling, subdivision, sharp edge retaining, surface smoothing, gaps bridging and haptic interface device.

Chapter 3 describes the proposed haptic-based hole filling algorithm for repairing defective holes.

Chapter 4 introduces the proposed sharp edge retaining and gap bridging algorithm for repairing incomplete surface models with defective holes around sharp edge, and gaps between two surfaces.

Chapter 5 shows illustrative examples and results of the proposed computational techniques and algorithms.

Chapter 6 presents the conclusions and future research.

Chapter 2

Literature Review

In this chapter, we review the previous researches in 3D virtual model reconstruction algorithm which is, as described by its name, used to create surfaces representing an object from point cloud data. The recent researches on hole filling, triangle subdivision and surface smoothing algorithm are firstly discussed. Previous researches on sharp edge retaining algorithm is discussed next. And lastly, we end this chapter with review of haptic interface device. These issues will be discussed further in the following chapters.

2.1 Virtual Model Reconstruction

A number of researchers have worked on modeling 3D virtual object from point cloud data. The developed object is usually illustrated in the form of triangular meshes. More information on recent researches can be found in [Mencl 1998] and [Bernaradini 1999]. Virtual object reconstruction algorithms can be classified into three major categories [Lin 2004] including sculpting-based, contour tracing and region growing approaches.

The sculpting-based approach includes alpha-shape [Guo 1997], γ - graph [Veltkamp 1995], β - skeleton [Kirkpatrick 1985], crust algorithm [Amenta 1998, 1999] and umbrella filter algorithm [Adamy 1999]. This technique builds surface of an object by first decomposing the space into cells using Delaunay tetrahedrization/Delaunay triangulation and then eliminating the cells that do not meet its criteria. Finally, triangulation is used to create triangular patches representing surface of the object. The approach is quite robust but it could suffer from computational difficulty and expensiveness, especially in the early step where structure of the space is set up.

[Curless 1996], [Bernardini 1997] and [Boissonat 2000] are examples of the second category, contour tracing approach. This method produces surface of an object using

approximation techniques, which might not accurately describe surface characterizations. Therefore, in some cases where accuracy is highly important e.g. in industrial product design or quality control purpose, contour tracing-based reconstruction might not be preferable.

The last category, region growing approach, begins by finding a starting triangle called seed triangle. Then, new triangles are added to each edge of the seed and newly created triangular patches. The process is continuously repeated until no point left behind and every point belongs to at least one triangle. The examples of region growing approach include ball point algorithm (BPA) [Bernardini 1999], [Huang 2002] and IPD algorithm [Lin 2004]. This method usually has a specific user-defined criterion for searching and creating new triangle. Thus, if the input point cloud is not dense enough, it could leave some defective holes and gaps in the model.

What we have reviewed so far is the brief summary of available surface model construction methods which might generate incomplete or defective model. We will show how our proposed algorithms deal with defective holes and gaps problem later in this paper. In the next section, we will review some of the previous researches in hole and gap filling, subdivision and surface smoothing algorithm.

2.2 Hole and Gap Filling, Subdivision and Surface Smoothing

Hole filling algorithm is typically used to repair the incomplete surfaces (e.g. missing triangle patches) caused by reconstruction process, while triangle subdivision and surface smoothing are used as post-processing techniques to improve surface quality. Loss of surface details usually appears in CAD model as defective gaps and holes. Holes and gaps problem is one of the problems that could be found in data acquisition process in Reverse Engineering [Varady 1997]. The most common approach that could be used to solve the problem of holes and gaps is to create new triangles and fit them onto the target area such that they seamlessly seal and conform to the curvature of boundary surface. This process is called hole and gaps filling.

The problem of detecting and repairing defective gaps in CAD object has been noticed since early of 90's. Berequet et al. (1995) proposed an algorithm that used a partial curve matching technique to match defective parts. Unmatched parts of the object can be solved using optimal triangulation of 3D polygon. This approach involved NP-hard problem and can only handle NURBS surfaces. Yau et al. (2003) introduced an extension of a surface reconstruction algorithm to global stitching of STL model. Since the approach deals with point data globally, it might consume lots of computational resources. Most of previous works in gap repair problem performs their algorithms automatically. Gaps detection relies heavily on the gap searching strategy and might not produce desired result. In this thesis, we present a technique to manually repair triangular patches based on stitching. The user can control the shape of created triangular patches and limit gap repair work at only desired area. Our method is also simple and intuitive. An example of gaps repair can be found in Figure 2.1.

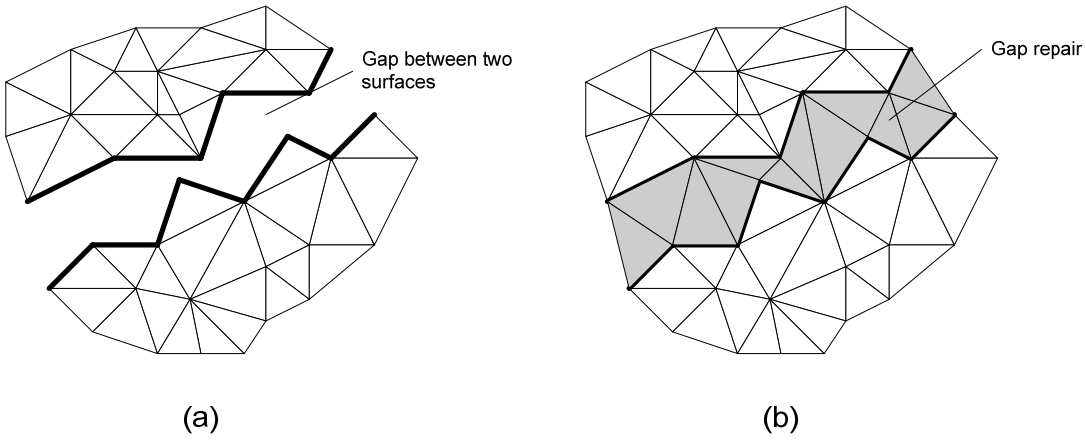


Figure 2.1 Gaps repair. (a) Example of a surface with gaps. (b) Gap after repaired.

In case of defective holes, the problems mostly encountered and must be solved by hole filling algorithm are triangulation over empty space and approximation of reconstructed surfaces such that they conform to the curvature of hole boundary. Previous

researches presented several approaches to repair defective holes in virtual object models. The first approach creates small triangles to fill holes in the first place. It may start by (1) growing a triangle from a hole boundary edge and progressively adding triangles around the hole perimeter until completely covering the holes by a set of triangles [Tekumalla 2004]; or (2) generating a set of points onto the holes and from each edge of hole boundary, beginning the search in that set of points for the most appropriated point to form new triangles [Wang 2003].

The second approach creates bulky triangles to fit the hole, and subdivides them into smaller triangles to increase resolution of the surface (so called subdivision algorithm e.g. [Karbacher 2001]). Examples of algorithms that use this approach are [Pfeifie 1996], [Liepa 2003] and [Jun 2005].

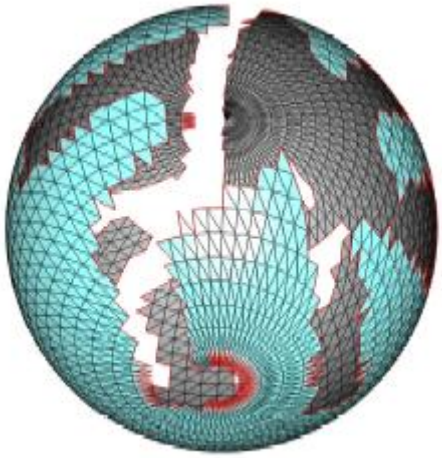


Figure 2.2 A sphere model with defective holes [Liepa 2003].

The third approach applies energy minimization function to repair the polygonal holes using cubic triangular spline patches [Chui 2000]. Another method fits the hole by a smooth surface according to the radial basis function interpolation method [Chen 2005]. Examples of holes and its repaired surface are shown in Figures 2.2 and 2.3 respectively. Most of previous approaches solve the problem based on the assumption that the

repairing algorithm should be performed to the whole model. In other words, every hole that appears in the model, regardless of its functions, will be repaired. However, this assumption may not be always true since in some cases the model might be required to have holes e.g. functional holes. Thus, in the problems where holes are required to remain in the models, the existing approaches might not be preferable.

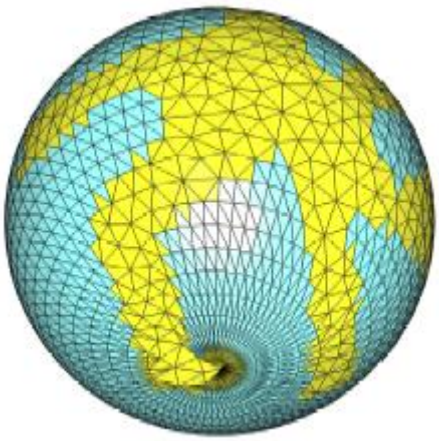


Figure 2.3 The repaired sphere model [Liepa 2003]

Regardless of which approach to be used, hole filling algorithm is often followed by subdivision and surface smoothing process to improve and ensure that the newly created surface best approximate the shape of the prototype. Subdivision (e.g., [Karbacher 2001]) is a process used to split a triangle into pieces in order to increase resolution of the repaired facets. It starts with big triangles and progressively divides those triangles into smaller pieces. This process is useful since the more triangles, the more accurate the surface could be. Readers can refer to [Ma 2005] and class note provided by [Bechmann 2006] for more information on review of subdivision algorithms.

After subdivision, the repaired triangles are checked if one or more of their vertices could be modified such that the surface looks more pleasant. This process is called surface smoothing or fairing. Wang et al. (2003) and Tekumalla et al. (2004) applied

surface smoothing into hole filling stage. They used moving least square technique to interpolate original data and reconstruct surface patches over the hole. Jun et al. (2005) and Karbacher et al. (2001) smoothed the surface during subdivision stage by first splitting the triangles and moving new vertices to new positions. Liepa et al. (2003) calculated and used umbrella operator to smooth irregular meshes.

In this paper, a new technique is proposed to provide choices to users such that they can decide which hole should be repaired and which hole should not, thus make it applicable to both situations. In addition, an extra system, haptic interface device, will be required for user intuitive hole selection. We will discuss with more details about haptic interface device later in the Section 2.4.

2.3 Sharp Edge Retaining

Defective hole problem becomes more complicated and difficult to solve when the hole appears at sharp edge of the model. Simple hole repair algorithms cannot be used in this case since they do not provide enough information to reproduce such feature. There are some existing researches in this specific problem and we will review those in this section.

A number of previous researches proposed algorithms to retain sharp edge in different manners. Lu et al. (2002) introduced a feature preserving haptic modeling system that applied extended marching cube algorithm to retain sharp edge during haptic-based sculpting. Attene et al (2003) proposed an approach to detect chamfer edges, subdivide, input and force new vertices to lie on intersection of planes to create sharp features. Ho et al. (2004) introduced extended marching cube algorithm to preserve sharp features of 3D model. The algorithm, based on feature-sensitive volumetric sculpting, recovers sharp features during dynamic editing of the model. Chen et al (2005) presented a sharpness dependent hole filling approach that is able to fill the hole of mesh-based model and recover sharp edge located at the hole area. Fleishman et al. (2005) proposed a robust least square technique which is able to reconstruct a piecewise smooth surface with sharp

edge retained.

Although these approaches describe different techniques to attack the problem, they have one thing in common. That is, they rely heavily on the similar assumption that sharp edge is located where surfaces have strongly different normal vectors (e.g. in industrial part where normal vectors of the surfaces are quite different, as shown in Figure 1.4). This assumption may not be true all the time, especially in free-form models as shown in Figure 1.5 where the normal vectors are not much different.

In this paper, the proposed algorithm is based on the fact that the sharp edge retaining algorithm should not limit its function to only geometric or industrial parts. It should also be able to help create sharp edges in free-form model as well. To provide this ability to sharp edge retaining, we needs to have user manipulation. An extra tool, which in our case is a haptic interface device, is required. The device allows users to observe and gain information of the surface of an object by providing force feedback. Users can feel the part as if they are touching it. Haptic interface device, therefore, helps users make decision to which part of the model that the sharp edge retaining algorithm should be performed. We will give briefly review about this specific tool in Section 2.4.

2.4 Haptic Interface Device

Haptic is a term from the Greek “haptesthai” which means “to touch” [Otaduy, Lin 2005]. Haptic is defined as an adjective and is used to describe something based on the sense of touch. Haptic interface device gradually becomes an important part in virtual reality system since it allows us not only “to see” and “to hear”, but also “to touch” objects in the virtual world. Haptic rendering is the term commonly used to describe processes of computing and generating forces in response to user interactions with virtual objects [Salisbury 1995].

Haptic interface devices provide ways for human to interact with objects in virtual world [Salibury 2004]. They may be classified by their grounding location, intrinsic mechanical behavior or number of degree of freedoms (number of dimensions that the

device can make movements). Example of different types of haptic interface device is illustrated in Figure 2.4. With these devices, there have been several related researches in different areas e.g. virtual sculpting, object design and prototyping, surgical simulation, training and practice, and etc. Figure 2.5 illustrates our lab setup haptic device.

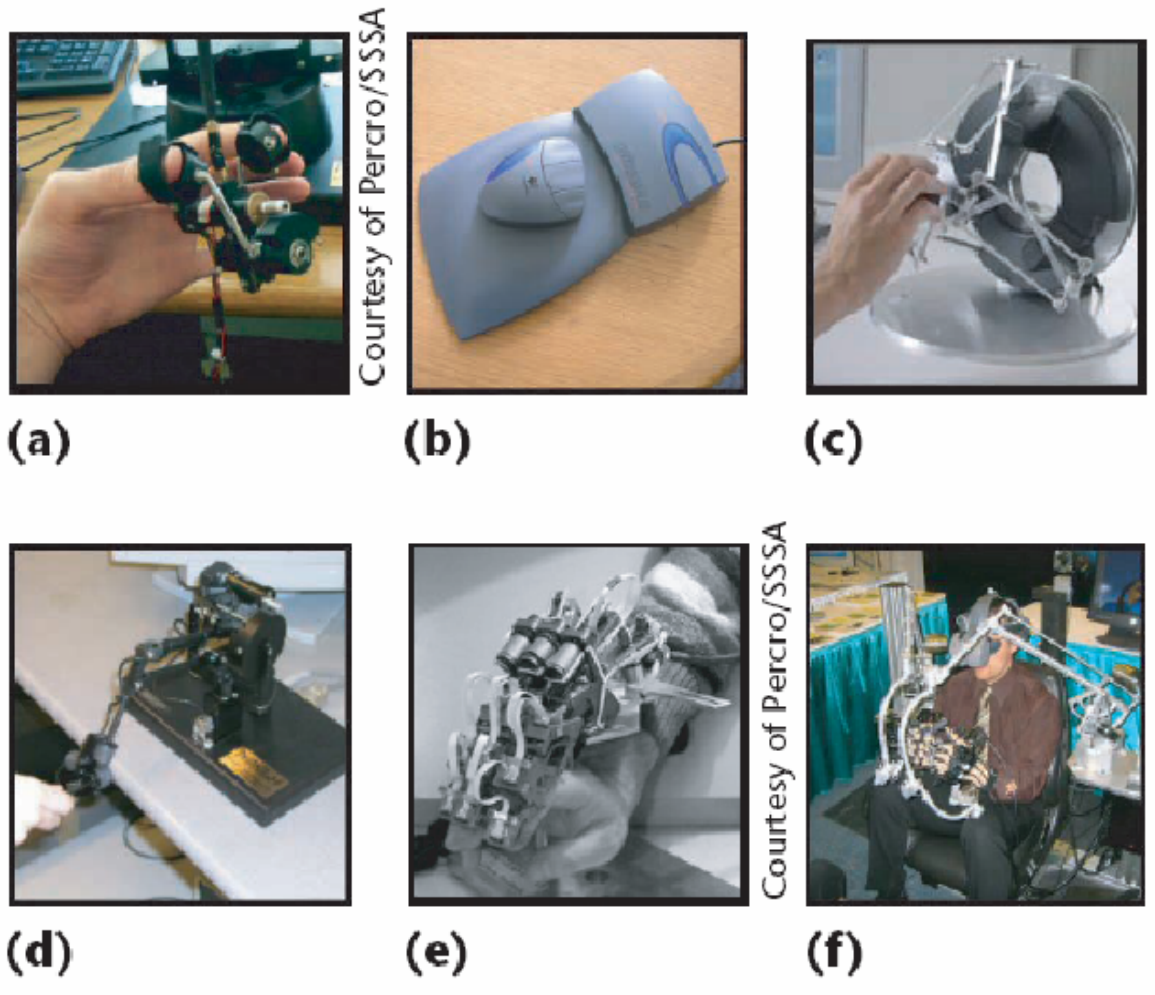


Figure 2.4 Examples of haptic interface devices [Salisbury 2004]: (a) force-reflecting gripper, (b) Force-feedback mouse, (c) Omega haptic device, (d) Phantom haptic device, (e) Hand Force Feedback exoskeleton, and (f) Haptic Workstation.

Members of our research group have worked with lab setup haptic interface device and applied its abilities to study some research works as show by following:

1. Energy-Field Optimization and Haptic-Based Molecular Docking and Assembly Search System [Lai-Yuen 2006].
2. Energy-Field Optimization and Haptic-Based Molecular Docking and Assembly Search System for Computer-Aided Molecular Design (CAMD) [Lai-Yuen 2005]
3. Five-axis Pencil-Cut Machining Planning and Virtual Prototyping with a 5-DOF Haptic Interface [Zhu 2004].
4. Dexel-Based Force-Torque Rendering and Volume Updating for 5-DOF Haptic Product Prototyping and Virtual Sculpting [Zhu 2004].
5. A Marching Algorithm of Constructing Polyhedral Models from Dexel Models for Haptic Virtual Sculpting [Zhu 2005].
6. Virtual prototyping and manufacturing planning by using tri-dexel models and haptic force feedback [Ren 2006].

Figure 2.5 Phantom Desktop™ Haptic Device used in our research.

Early researches used fully-automated approaches to solve sharp edge retaining problem. They depended solely on information and computational strategies of each algorithm, thus made them inflexible and unintuitive. Our method, which is presented in a working paper [Boonma 2006], combines computational techniques and user-interface system through haptic interface device. As a result, users have more freedom and ability to custom-make sharp edge. Our technique is also more flexible and intuitive since users can see the sharp edge during working and they can decide whether they are satisfied with the result or they want to re-specify the parameters and re-create sharp edge again until satisfied.

2.5 Summary

In this chapter, we briefly review some of related techniques in 3D virtual object reconstruction, hole filling, subdivision, surface smoothing, sharp edge retaining algorithm and haptic interface device. In the following chapter, haptic-based hole filling algorithm will be discussed.

Chapter 3

Haptic-Based Hole Filling Algorithm

This chapter presents an approach used to repair holes in 3D virtual object. By repairing, we mean to fill the holes with triangular meshes in order to create a watertight surface model. We focus on a specific hole problem where the defective holes are located at a sharp edge or parting line where two surfaces meet. Our goal is not only to repair the holes, but also to retain sharp edge the surfaces. To achieve the objectives, we propose two major steps. The first step is to automatically create triangular meshes over the hole and the second step, which will be discussed in the next chapter, is to use haptic interface device to detect and help re-creating sharp edge.

3.1 Introduction

In this chapter, we introduce haptic-based hole filling system. In our hole repairing, the input is an incomplete model with a defective hole locating at sharp edge of the object. Example of the incomplete model is shown in Figure 3.1. We begin our approach by applying an automatic hole search algorithm in order to detect the hole. Then, we calculate the hole normal vector, project the hole down onto a reference plane and roughly fill the hole with bulky triangles. Triangle subdivision is required afterward in order to increase the resolution of triangular patches and get rid of those bulky ones. The resulting surface is a watertight triangulate facet model.

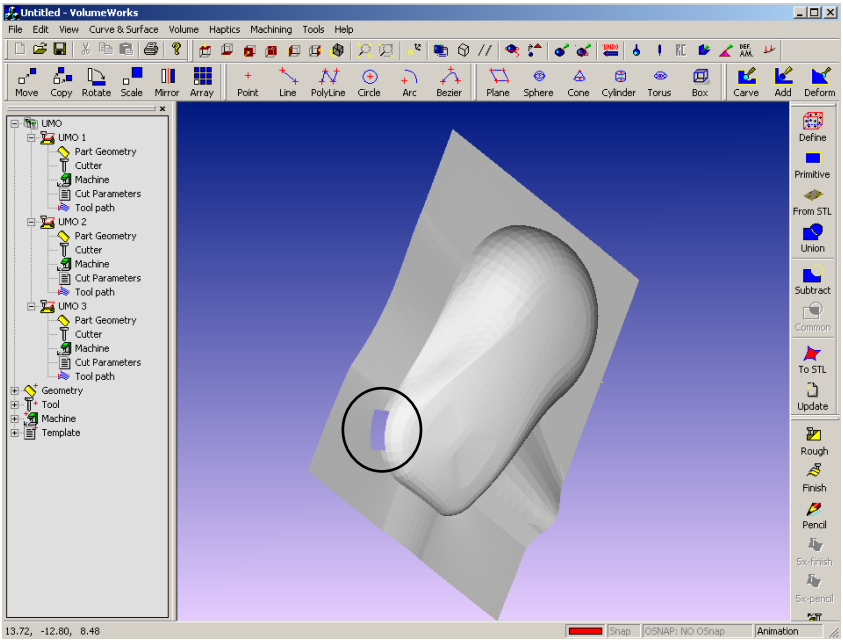


Figure 3.1 Example of a model with a defective hole locating at sharp edge

The remainder of this chapter is organized as follows: Section 3.2 presents haptic interface system for repairing defective surface model. Section 3.3 introduces hole search algorithm. Section 3.4 discusses haptic-based hole filling algorithm. Finally, section 3.5 provides the summary.

3.2 Haptic interface system for repairing defective surface model

Since we want to have a flexible technique that allows the user to selectively repair only the defects, the lab setup 6 degree of freedom (6-DOF) Phantom Desktop™ haptic device, as shown in Figure 3.2, was integrated into our system to provide force torque feed back to the user and accessibility to the surface of the input model. In the proposed technique, the haptic device is used in 3 primary functions: (1) to measure the reaction force fed back to users, (2) to measure and locate the positions of any points in three dimensional space and (3) to facilitate selection of particular objects, points and surfaces in 3D space.

We use surface rendering technique rather than volumetric rendering because it is faster and convey sufficient information required for performing the repair operations. Contact point where the virtual tool touches surface of the objects is used for haptic rendering since it is simple and consumes less computational resources.

In this paper, we will use the following notations: (1) virtual tool shown on the screen represents the point that haptic device is currently at, (2) haptic probe is the handheld input device moved and operated by the users.

Figure 3.2 Our lab setup haptic interface device.

In the next section, we will discuss the preliminary step that must be executed before using hole filling algorithm. This pre-process is called hole search.

3.3 Hole Search Algorithm for locating holes in surface models

In this section, hole search approach that is used to automatically locate holes in the input model is discussed. We employ hole search algorithm introduced by [Chua 2003] in our work. The algorithm consists of two major parts. The first part searches for all boundary edges which have only one triangle adjacent to it. Then the second part of the algorithm sorts all boundary edges obtained from the first part and forms them into a loop. Hole search algorithm must be performed according to this order, boundary edge search and then edge sorting.

The boundary edge search is simple and can be described by the following steps in reference to Figure 3.3. Boundary edges are also displayed as the darken line in the Figure.

1. For each edge e_{ij} in the model, search for an edge e_{ji} which is in the opposite direction of the edge e_{ij}.
2. If edge e_{ji} does not exist, save it in a list.
3. Repeat step 1 until all edges in the model are considered.

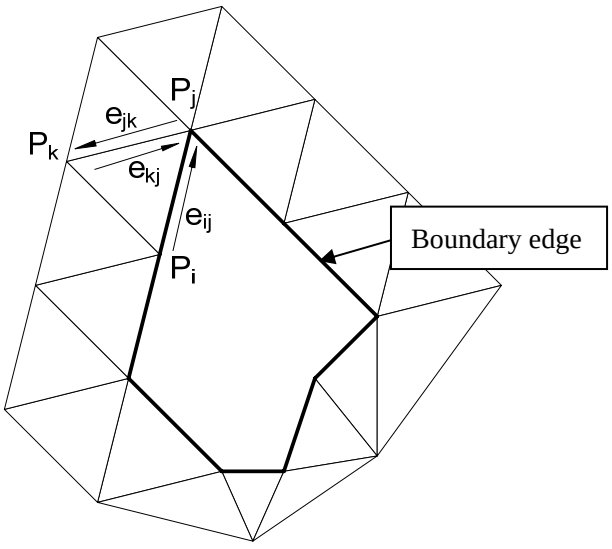


Figure 3.3 Boundary edges (shown by the darken line). Edge e_{ij} does not have its reverse pair (e_{ji}), thus is classified as a boundary edge.

After we get all the boundary edges, start sorting these edges such that they are consecutively connected and form a loop shape (first point of the first edge is the same as end point of the last edge). The edge sorting, as shown in Figure 3.4 can be described by the following steps.

1. Get a boundary edge e_{ij} from the list created during boundary edge search and put it in a temporary list.
2. Search in the boundary edge list for an edge whose starting point is point P_j. If such edge is found, put it at the end of the temporary list.
3. In the temporary list, test if the end point of the last edge is the starting point (P_i) of the first edge. If the result of the test is TRUE, set all edges in the temporary list as a boundary of the hole, clear all edges in the temporary list and repeat step 1 in case there are still some edges left in the boundary edge list. Otherwise, repeat step 2.

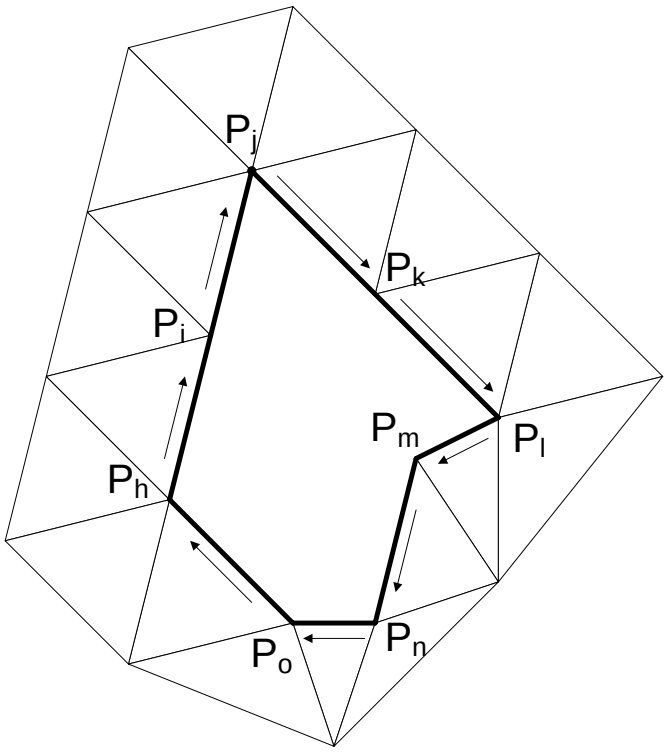


Figure 3.4 Edge sorting forms a hole which, in this case does not have any coincident points.

The edge sorting was constructed such that it can handle the situation where holes do not have any coincident point as illustrated in Figure 3.4. We don't consider the case where two or more holes have a coincident point as shown in Figure 3.5. Solution to such situation can be found in [Chua 2003].

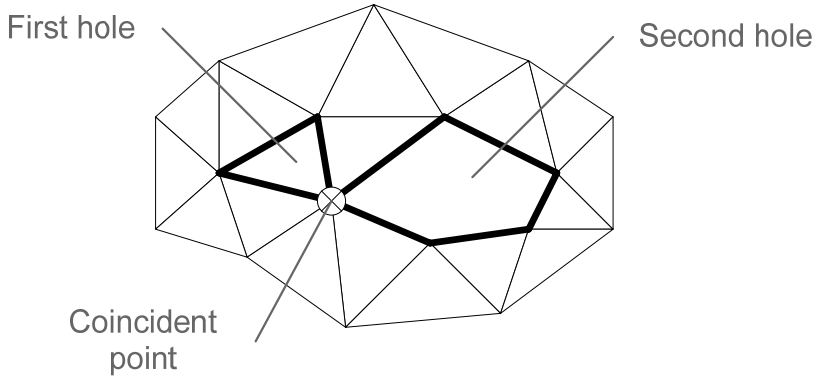


Figure 3.5 Holes with a coincident point.

After hole search, we have a list of holes containing loops of connected edges. Each loop represents a hole in the model. Our program allows the user to select a hole that is to be repaired. The user can select the hole by first moving the virtual tool to the location where the desired hole is located. When the tool is close enough to the hole, the hole will be highlighted automatically to let the user know which hole he/she is going to work on. Then the user can presses the button on the haptic probe while the probe is making contact with a surface near the hole boundary. This confirmation will begin the hole filling algorithm. Details of hole filling algorithm will be discussed next in Section 3.4.

3.4 Haptic-Based Hole Filling Algorithm for repairing defective holes

After user makes selection, the chosen hole is automatically repaired using hole filling algorithm and followed by triangle subdivision. Surface smoothing algorithm will be done after sharp edge retaining process. Major steps of our hole filling algorithm can be shown in Figure 3.6.

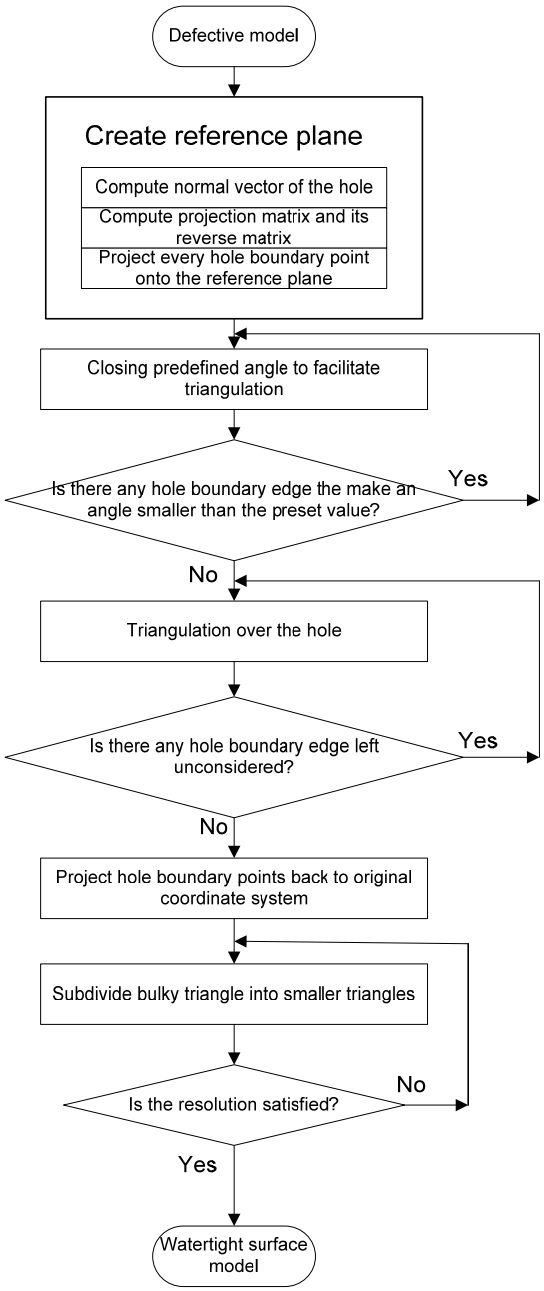


Figure 3.6 Flow chart of **Algorithm I: haptic-based hole filling algorithm**

Our algorithm shown above is simple and straightforward. At the end, we obtain the hole filled with un-oriented triangles which will be improved later by surface smoothing technique. Let us describe more details of each step involved in our hole filling algorithm, beginning with the computation of hole normal vector, rotational matrix and reference plane set up. These early steps are discussed in section 3.4.1.

3.4.1 Creating Reference Plane for boundary points projection

We decided to create triangle patches to cover the hole in 2D since it is rather easy to do the computations. Also, a simple intersection test between the new created triangle and the hole boundary can be used. What we need to do first, is to transform the hole coordinate system into 2D by projecting all hole boundary points onto a plane. This plane is called reference plane. To create the reference plane, the hole normal vector, which will be used as normal vector of the reference plane, is calculated by averaging normal vectors of surrounding triangles (hole boundary triangles) [Jun 2005]. Illustrative example of hole normal vector calculation is found in Figure 3.7.

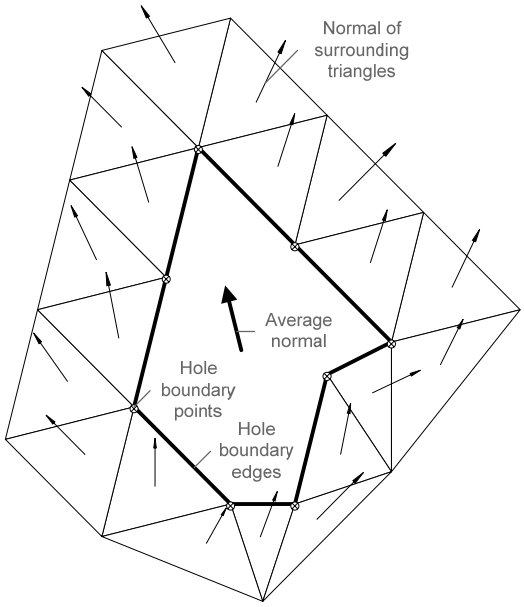


Figure 3.7 Hole normal vector. The vector is calculated by averaging all normal vectors of the boundary triangles.

After the reference plane is created, we project all hole boundary points onto the reference plane by calculating a rotational matrix. This matrix is meant to rotate the reference plane to the angle that normal vector of the reference plane points to positive Z direction. If we apply the same rotational matrix to all hole boundary points, their projections on the reference plane are simply the X and Y coordinates [Jun 2005]. Up to this point, we have the hole boundary lying on a 2D plane. What we want to do next is to eliminate connected boundary edges that make the angles smaller than a predefined value. The "closing predefined angle" will be discussed next, in section 3.4.2.

3.4.2 Closing Predefined Angle to facilitate triangulation

After we have the defective hole lying on a 2D plane, the next step is to eliminate the angle between two boundary edges that is smaller than a predefined value. We do this by moving around the hole perimeter and obtain two consecutive edges. From these two edges, e.g. edge e_{ij} and e_{jk}, we calculate the angle between them and check whether the angle is smaller than a predefined value. If the angle is indeed smaller, we then perform intersection test if we create new triangle $\Delta P_i P_k P_j$ and replace edge e_{ij} and e_{jk} by edge e_{ik}. The test is to check if the newly created triangle intersects with any of the hole boundary edges, as shown in Figure 3.8. If such intersection is detected, triangle $\Delta P_i P_k P_j$ cannot be created and the search moves over to check the next two connected edges. On the other hand, if no intersection is found as shown in Figure 3.9, we create triangle $\Delta P_i P_k P_j$ and replace edge e_{ij} and e_{jk} by edge e_{ik}. Repeat the steps until no small angle between any two consecutive edges is detected or the hole is completely filled with newly created triangles. After testing with some values, setting the angle to $\frac{5\pi}{9}$ gave satisfied result [Tekumalla 2004].

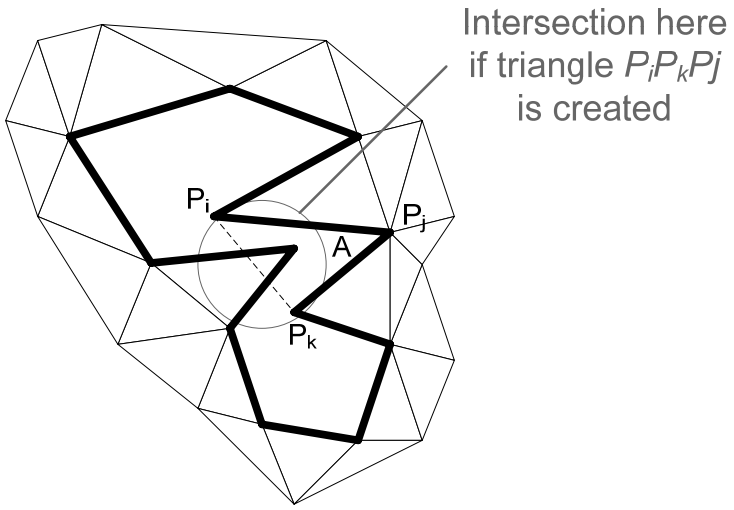


Figure 3.8 Intersection of $\Delta P_iP_kP_j$ and other existing triangles in the model.

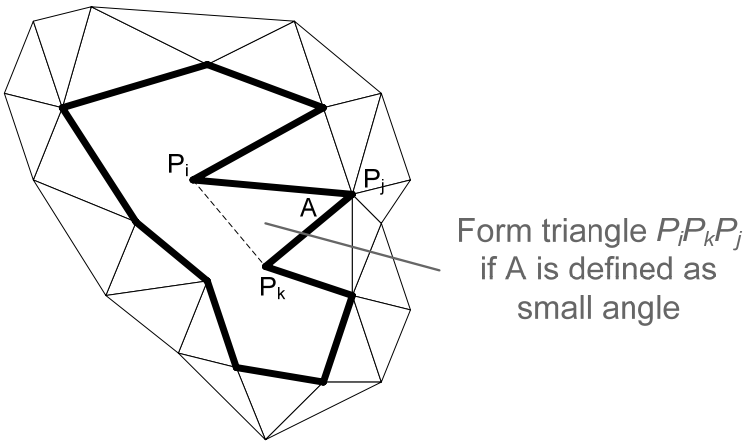


Figure 3.9 No intersection between triangle $P_iP_kP_j$ and other existing triangles in the model.

Our selected hole at this stage is simpler to be repaired since we already eliminated the angles that are smaller than the preset value along the hole boundary edges. In the next section, we will fill the hole with large triangles.

3.4.3 Triangulation over the hole area

After closing small angles, we do triangulation over the hole area in order to fill the hole with triangles. Basic idea of this step is for each hole boundary edge e_{ij}, we search for another hole boundary point P_k that is closest to both point P_i and P_j. The new triangle $\Delta P_iP_jP_k$ created from the closest point P_k must not intersect with any hole boundary edges as shown in Figure 3.10. If it intersects, we cannot create this triangle and have to look for another possible point. The algorithm, as described below, in the form of pseudo code will be repeated and creates new triangle along the hole boundary edges until the hole is completely filled.

Algorithm_II. Triangulation over the hole

Input: a hole represented by hole boundary edges

Output: filled hole with bulky triangles

```
WHILE (hole is not completely filled)
{
    SAVE all hole boundary point into buffer point list
    REPEAT
    {
        GET an edge e_ij from hole boundary edges list
        FOR (all points in buffer point list)
        {
            COMPUTE square distance between a point in buffer point list and first and
            end point of the current edge
        }
        GET the point P_k that has minimum sum of square distances
        /* Check whether new triangle ΔP_iP_jP_k can be created. Test passes if no
        intersection is detected */
        INTERSECTION_TEST
        IF (no intersection detected)
        {
            CREATE triangle ΔP_iP_jP_k and edge e_jk and e_ki
            /* Update hole boundary edges */
            REPLACE edge e_ij by edge e_jk and e_ki
        }
        ELSE
```

{

DELETE point P_k from buffer point list

}

} **UNTIL** (no point left in buffer point list or new triangle $\Delta P_iP_jP_k$ can be created)

}

END

In INTERSECTION_TEST, we check whether the to-be-created triangle intersects with any existing hole boundary edges. If it does, the to-be-created triangle must not be constructed. We then skip the current edge and obtain the next one. On the other hand, if the to-be-created triangle does not show any intersection with any of the hole boundary edges, we are free to construct the triangle. Figure 3.10 shows that triangle $\Delta P_iP_jP_k$, which passes the intersection test (it does not intersect with any of the hole boundary edges), is created, while the triangle $\Delta P_iP_mP_j$ must not be created since it shows intersection with one of the other hole boundary edges. The algorithm will be repeated until the hole is completely filled as shown in Figure 3.11.

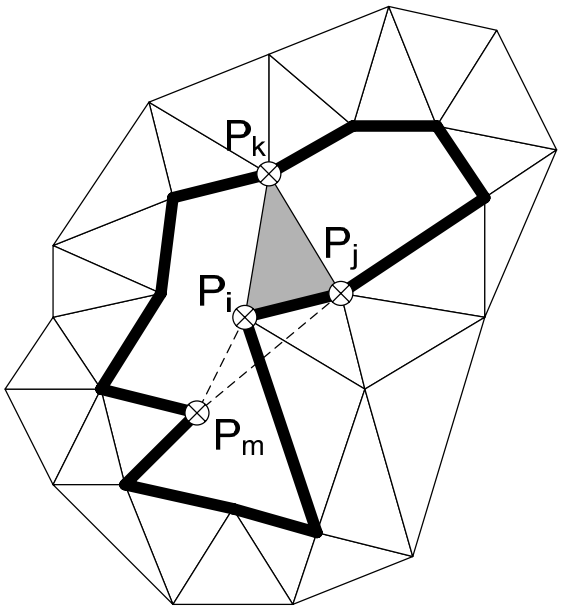


Figure 3.10 Unlike triangle $\Delta P_iP_mP_j$, triangle $\Delta P_iP_jP_k$ shows no intersection and thus can be created.

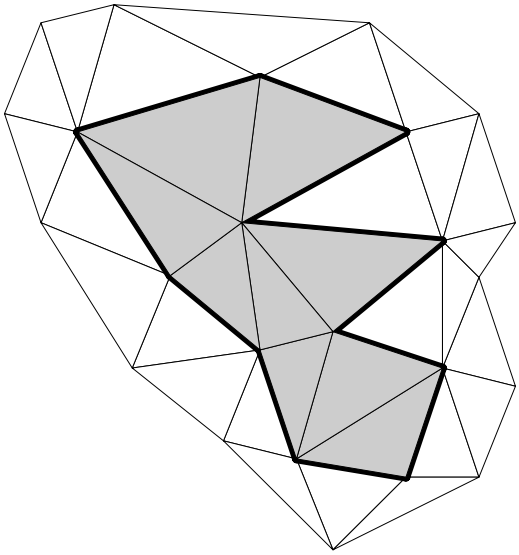


Figure 3.11 The hole filled by Triangulation.

After completely filling the hole, as displayed in Figure 3.11, we multiply all of the projected hole boundary points with the reverse rotational matrix in order to turn them back into the original coordinate system. Up to this point, our incomplete model is refilled with large triangle and becomes a watertight one.

In order to increase resolution and improve accuracy of the newly created triangle patches, in the next process, we will subdivide those triangle patches into smaller pieces. This upcoming process is called subdivision.

3.4.4 Subdivision

Subdivision process is important and must be done as a part of our approach in order to improve resolution and accuracy of the model. However, it should not create too many triangles since large number of triangles might slow the overall computation. Our suggestion is to create triangles such that their sizes are close to hole boundary triangles in the model. This can be done by pre-calculating the average length of those hole boundary triangles and setting a preset value for subdivision algorithm such that, when

we reach the predefined value, the algorithm will stop. We employ refinement algorithm introduced by [Liepa 2003] in our work and the steps involved in the algorithm are as shown by following.

Algorithm_III. Subdivision

Input: a triangulate facet model with bulky repaired triangles

Output: a triangulate facet model with smaller repaired triangles

DO
{
COUNT number of times that the algorithm runs this loop

SET starting triangle to the first triangle in the list, e.g., $\Delta P_i P_j P_k$

WHILE (at least one triangle has not been considered)
{
createNewTri=FALSE;
IF (the triangle being considered is already considered in this running loop)
{
CONTINUE /* go back and get another triangle */
}
COMPUTE average length of all edges connecting to each vertice of the triangle being considered, obtaining scale attributes σ_i, σ_j *and* σ_k

COMPUTE the average value σ_c by averaging σ_i, σ_j *and* σ_k

COMPUTE centroid of the triangle being considered (P_c)
COMPUTE distance between each vertice and the centroid, and multiply the each distance value by a square root, obtaining $dist_i$, $dist_j$ and $dist_k$

IF(($dist_i > \sigma_c$ and $dist_i > \sigma_i$) or ($dist_j > \sigma_c$ and $dist_j > \sigma_j$) or

($dist_i > \sigma_c$ and $dist_i > \sigma_k$))

{
/* subdivision */
SET createNewTri = TRUE
SUBDIVIDE the triangle being considered into 3 pieces;
CONTINUE; /* exit the loop */
}

```
    }

    IF( createNewTri = FALSE ) /* if no subdivision or no new tri created*/
    {
        /* mesh is complete */
        BREAK;
    }

    /* relax all interior edges of triangles*/
    DO
    {
        SET starting triangle to the first triangle in the list
        interiorEdgeSwap=FALSE;
        WHILE( at least one triangle has not been considered )
        {
            FOR(each edge of a trianalge)
            {
                IF(the edge is not boundary edge, and not considered before)
                {
                    /* check to relax this edge with its neighbor triangle */
                    IF( pass criterior to do interior edge swap )
                    {
                            interiorEdgeSwap=true;
                            break; // if relax this edge, consider next tri
                    }
                }
            }
        }
    } WHILE ( interiorEdgeSwap );
    /* if we swap any of the interior edge, check to do swap until no swap is found  */
} WHILE ( createNewTri && (count!=preset value) );
/* if we subdivide a triangle, keep do it until no subdivision occur. */
END
```

Relaxing method is required to obtain better shape triangle which usually has the same length for all three edges. Long and narrow triangle is not preferred in CAD works since it tends to provide a model that is difficult to be modified. In relaxing method, the triangle $\Delta P_iP_jP_k$ is replaced by triangle $P_iP_jP_c$, $P_jP_kP_c$ and $P_kP_iP_c$ as shown in Figure 3.12. Then, edges e_{ij}, e_{jk} and e_{ki} are determined whether they can be swapped. The test can be described as follows. For example, triangle $\Delta P_jP_kP_i$ and $\Delta P_iP_mP_j$ have a

shared edge e_{ij}. Edge e_{ij} can be swapped if vertex P_k is inside the circum-sphere of the triangle $\Delta P_i P_m P_j$ as shown in Figure 3.13. This edge swap process is called edge relaxing [Liepa 2003].

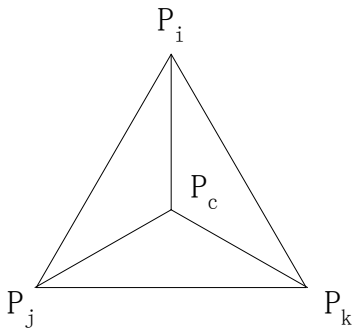


Figure 3.12 Splitting Triangle. Triangle $\Delta P_i P_j P_k$ is replaced by triangle $P_i P_j P_c$, $P_j P_k P_c$ and $P_k P_i P_c$

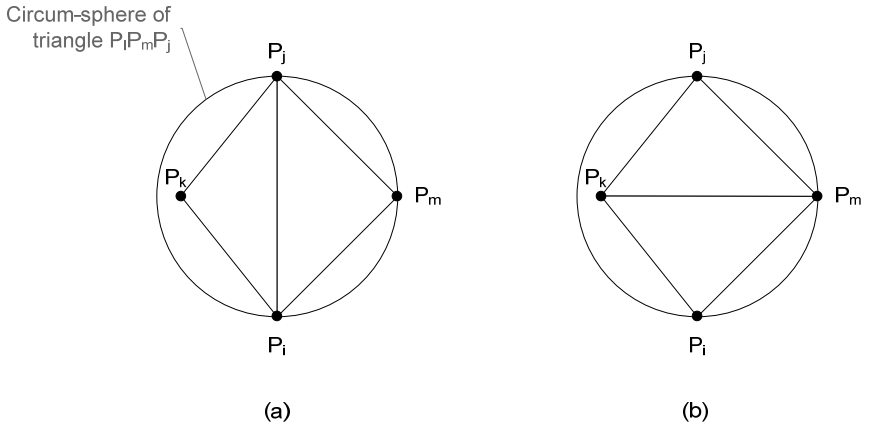


Figure 3.13 shows edge relaxing method. (a) before relaxing edge e_{ij}, point P_k is inside the circum-sphere of triangle $\Delta P_i P_m P_j$. (b) after relaxing, edge e_{ij} becomes edge e_{km}.

The difference between our work and [Liepa 2003] is that we set a preset value such that when the repetition reaches the setting number, subdivision will be stopped. The preset value is determined by the user. In our experiment, the number ranging from 1 – 5

provides satisfied result. This setting could speed up the computation and limit number of triangles being produced.

The hole is now filled up with smaller triangles as we would like it to be. These triangles will be called repaired or hole triangles throughout the rest of this paper. After subdivision process, if the hole is not located at the sharp edge, we can apply surface smoothing algorithm and we should be able to get watertight model with better surfaces that approximate the curvature of hole boundary triangles.

3.5 Summary

In this chapter, we have presented the haptic-based hole filling for repairing defective holes. The problem becomes more interesting if hole is located right at the sharp edge of the part. In such case, hole filling algorithm alone is not sufficient to repair the incomplete model. We describe our solution to this specific problem in the following chapter.

Chapter 4

Haptic-Based Sharp Edge Retaining and Gap Bridging Algorithm

This chapter presents two approaches: (1) haptic-based sharp edge retaining algorithm for creating sharp edge over defective holes, (2) haptic-based gap bridging algorithm for sealing the gap between two triangulated surfaces. For the first technique, we focus on a specific problem where a defective hole is located at a sharp edge of the part. Our input is a watertight model which is the result of the hole filling algorithm as discussed in previous chapter.

4.1 Introduction

For an incomplete model that a defective hole lies on its sharp edge, we first use hole filling algorithm that we presented in the previous chapter to re-create surface patches over the hole (repaired triangles). Then, sharp edge retaining algorithm is required to recover the sharp edge at the hole area.

With a hole-filled model as our input, haptic interface device is used to specify four points along the sharp edge on the hole boundary surface. These points are inputs for our algorithm in order to create a Hermite curve approximating the sharp edge and the guiding surface across the hole. The repaired triangles intersected by the guiding surface are re-aligned according to the guiding surface and users can make adjustment to those triangles by moving them down either to the satisfied level or to the pre-calculated sharp edge (which is represented by the Hermite curve). The final process is to smooth the created triangles using simple surface smoothing algorithm.

In our gap bridging part, we don’t consider the problem of registration of two surfaces since we assume that the two surfaces are perfectly registered and ready to be bridged together. We just show our concept to do the gap bridging using haptic interface

device. The inputs of our system are two perfectly registered surfaces. The haptic interface device can be used to 1) manually create triangles such that they connect these two surfaces together or 2) manually create two triangles at both sides of the gap such that the gap turns into a hole. Then, the traditional hole filling algorithm is applied. The result of this algorithm is refilled gap and two surfaces are merged together.

The remainder of this chapter is organized as follows: Section 4.2 describes haptic-based sharp edge retaining algorithm. Section 4.3 introduces haptic-based gap bridging algorithm. Finally, section 4.4 provides the summary.

4.2 Haptic-Based Sharp Edge Retaining Algorithm

In this section we introduce haptic-based sharp edge retaining algorithm. Previous works create sharp edge if normal vectors of triangle patches representing the surfaces are obviously different. Those algorithms, therefore, are suitable for industrial parts where the information of sharp edge can be extracted from the hole boundary surfaces.

In our work, we would like to show that not only in the industrial parts that sharp edge can be created, but it also can be obtained in the free-form model where the shape of sharp edge is not well defined. We use haptic interface device to capture information of sharp edge from the hole boundary triangles. Then, the information is used to align the hole triangles in order to facilitate changing position of the triangles along the approximated sharp edge. To create sharp edge, the user can now either push these triangles down to their preferred level or to the approximated level that the algorithm calculates. Simple surface smoothing algorithm is performed next so that the triangles approximate the shape of the physical object. The major steps of our algorithm can be shown in Figure 4.1

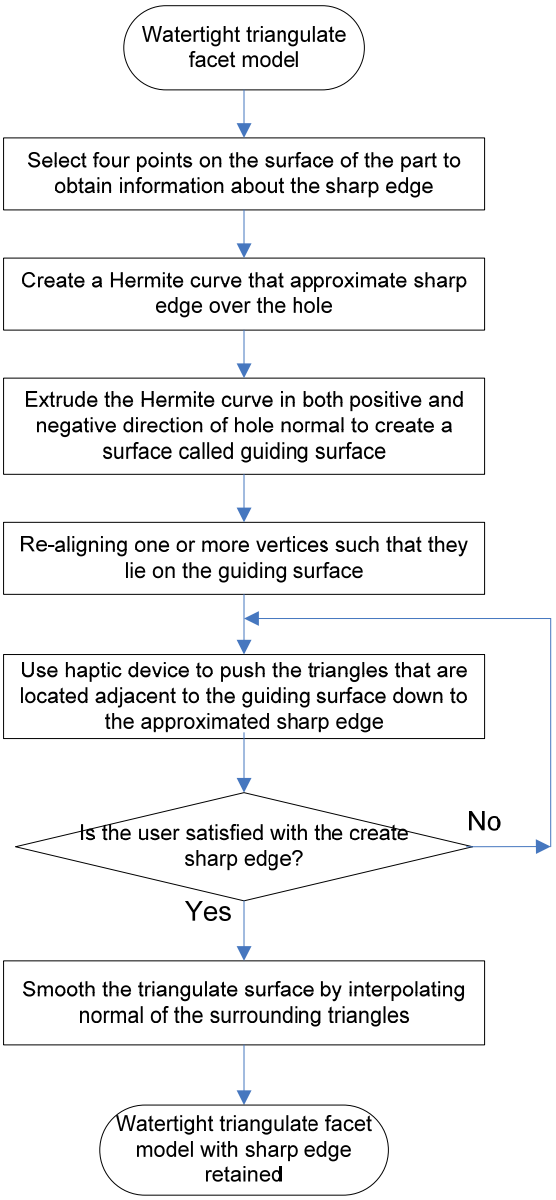


Figure 4.1 **Algorithm IV: haptic-based sharp edge retaining algorithm**

At current stage, our program can handle only sharp edge on concave surface since the sharp edge can be easily detected by the haptic interface device. Our algorithm begins at the time that the user uses haptic interface device to select four points along the sharp edge on the hole boundary triangular patches. Order of point selection can be illustrated in Figure 4.2. Then, these four points will be used to construct a Hermite curve to

approximate the sharp edge over the hole. The concept of Hermite curve can be described in the next section.

4.2.1 Constructing Hermite Curve and Guiding Surface

Hermite curve is a form of cubic polynomial curve [Foley 1996]. It can be defined by end point P_1, P_4 and the tangent vector R_1 and R_4 at end point P_1 and P_4 respectively. Hermite curve can be represented by the following equation.

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \bullet M_H \bullet G_H \tag{1}$$

Where G_H is the column vector $\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$.

M_H is Hermite basis matrix which is uniquely defined as $\begin{bmatrix} +2 & -2 & +1 & +1 \\ -3 & +3 & -2 & -1 \\ 0 & 0 & +1 & 0 \\ +1 & 0 & 0 & 0 \end{bmatrix}$

Length of tangent vector at R_1 and R_4 affects the shape of Hermite curve. The longer the length of tangent vectors, the greater their effects are to the curve [Foley 1996]. Thus, the geometry vector $\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$ must have the form $\begin{bmatrix} P_1 \\ P_4 \\ k_1 r_1 \\ k_4 r_4 \end{bmatrix}$, with k_1 and $k_4 > 0$. The variable k_1 and k_4 are determined based on experimental basis in order to get the curve that best satisfies the user. The example of a Hermite curve created from four selected points is shown in Figure 4.2.

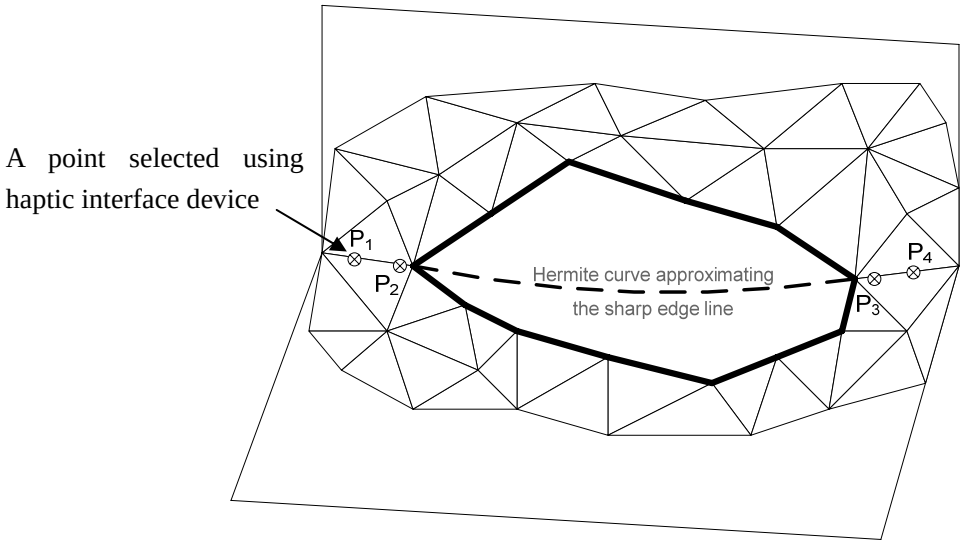


Figure 4.2 Example of four points (P_1-P_4) selected using haptic interface device, order of points selection and the constructed Hermite curve.

In our program, point P_2 and P_3 will respectively be our start and end point of the Hermite curve. Tangent vector r_1 at point P_2 is a unit vector from point P_1 to P_2, while the tangent vector r_4 at point P_3 is a unit vector from point P_3 to P_4. Depending on the example model we used, we set number of points on the Hermite curve such that each segment between points is approximately the length of edge of triangles around hole boundary. Number of points also represents number of line segment that we want to have in the Hermite curve. Also, the size of tangent vector can be defined using variables k_1 and k_4 as described in the previous section. We set the variables such that the resulted Hermite curve approximates the curvature of the model. This setting is based on experimental basis. Figure 4.3 shows Hermite curve approximating sharp edge on a mouse model.

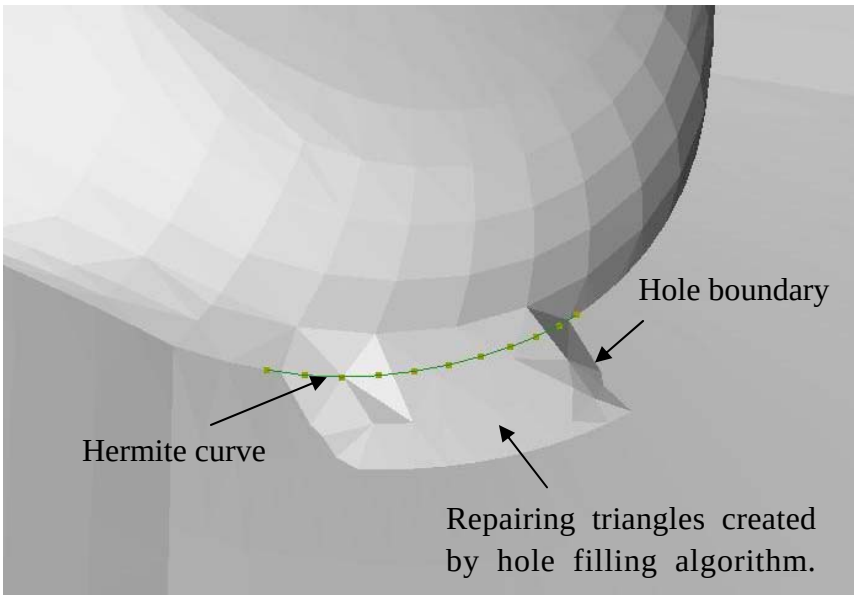


Figure 4.3 Hermite curve created from four selected points. The curve approximates sharp edge between bottom surfaces and the mouse.

After having the Hermite curve, we extrude it in the direction of hole normal vector and the direction opposite to hole normal vector. The extruded points are used in triangulation in order to create a guiding surface onto the Hermite curve as shown in Figure 4.4. Figure 4.5 illustrates the constructed guiding surface in a mouse model. The guiding surface, in later stage, will help the user to modify the reconstructed triangles through which the guiding surface passes.

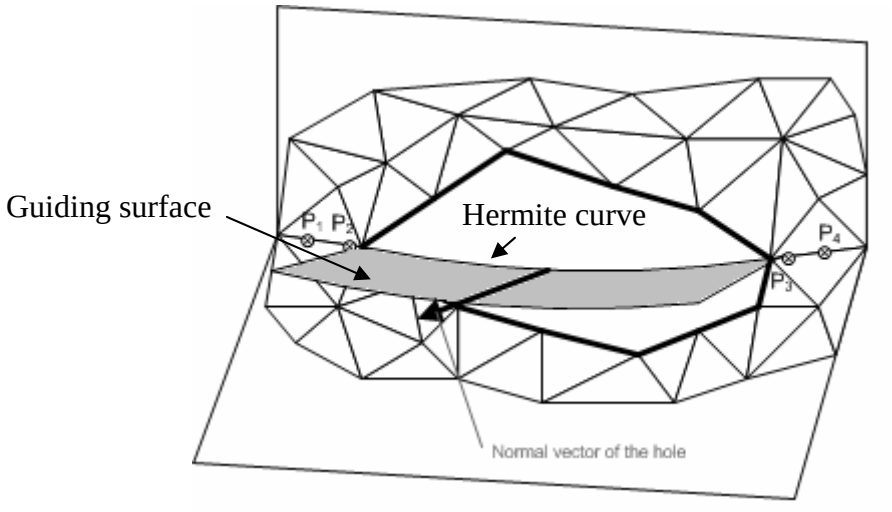


Figure 4.4 The guiding surface created from Hermite curve

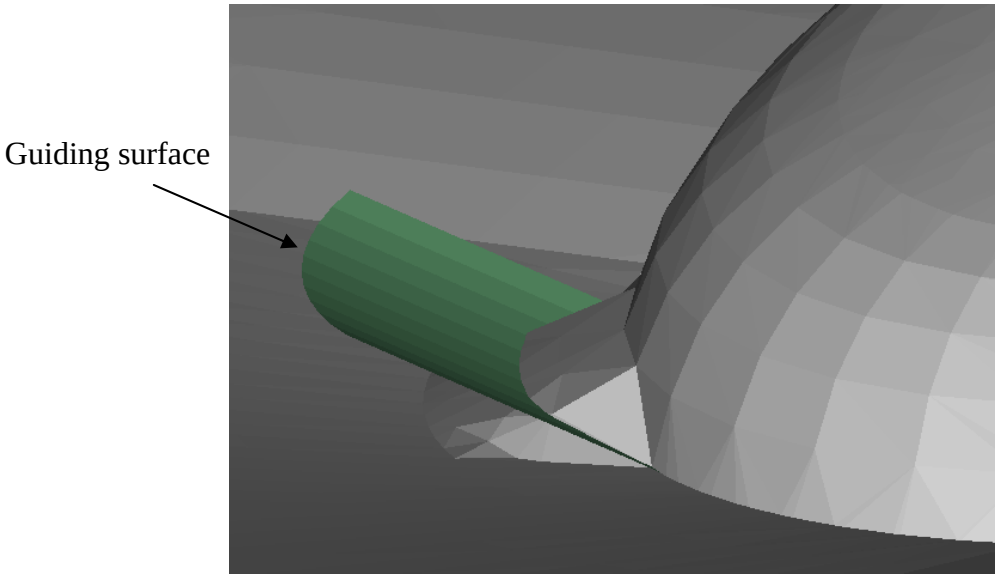


Figure 4.5 The guiding surface in the mouse model

All the hole triangles that the guiding surface passes through must be re-aligned so that they can be easily deformed by haptic interface device in order to create sharp edge. This topic will be discussed in section 4.2.2.

4.2.2 Vertices Snapping for re-aligning triangle vertices onto the guiding surface

In vertices snapping, we re-align some vertices of the triangles through which the guiding surface passes such that they are located on the guiding surface. Vertices snapping will facilitate surface deformation in the later stage where we use the haptic probe to push down the triangles along the guiding surface to the Hermite curve that approximates the sharp edge.

In this process, we search along the guiding surface for the repaired triangles through which the surface passes. Then, for each repaired triangle, we find a vertex that is located closest to the guiding surface and snap it onto the guiding surface. Vertices snapping can be done by following steps.

Algorithm_V. Vertices Snapping for re-aligning triangle vertices onto the guiding surface

Input: hole triangles

Output: aligned hole triangles

```
FOR (all hole triangles)
{
    FOR (all edges of the current triangle)
    {
        IF (the current edge intersects with the guiding surface)
        {
            GET the intersection point between the current edge and the guiding surface
            GET the closest end point to the intersection point and is located on this
            edge.
            IF (the end point we obtained is not hole boundary points)
            {
                SET the end point to the intersection point as shown in Figure 4.6.
            }
        }
    }
}
END
```

Figure 4.6 shows the alignment of hole triangles before and after being applied by our vertices snapping algorithm, while Figure 4.7 shows another example of our algorithm in the mouse model.

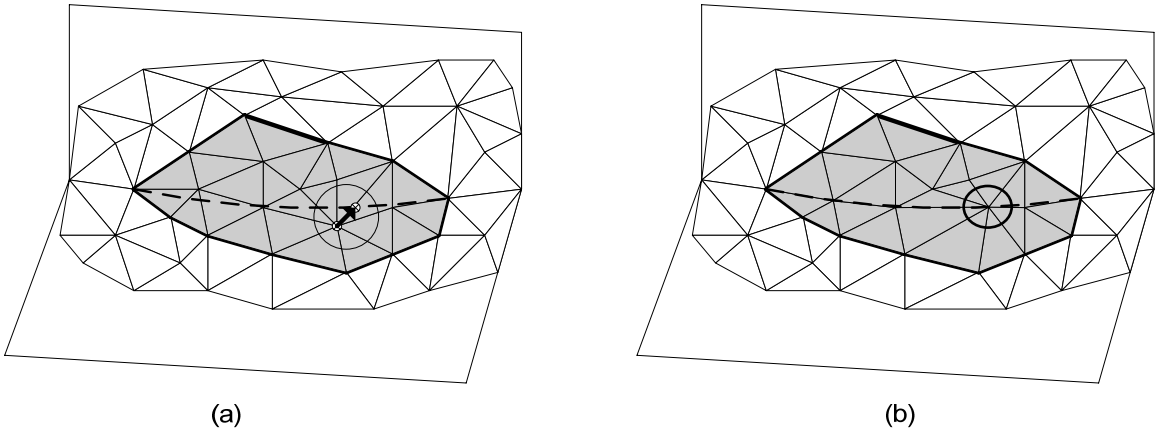


Figure 4.6 Vertex snapping. (a) Triangular meshes before vertex snapping. (b) Triangular meshes after vertex snapping.

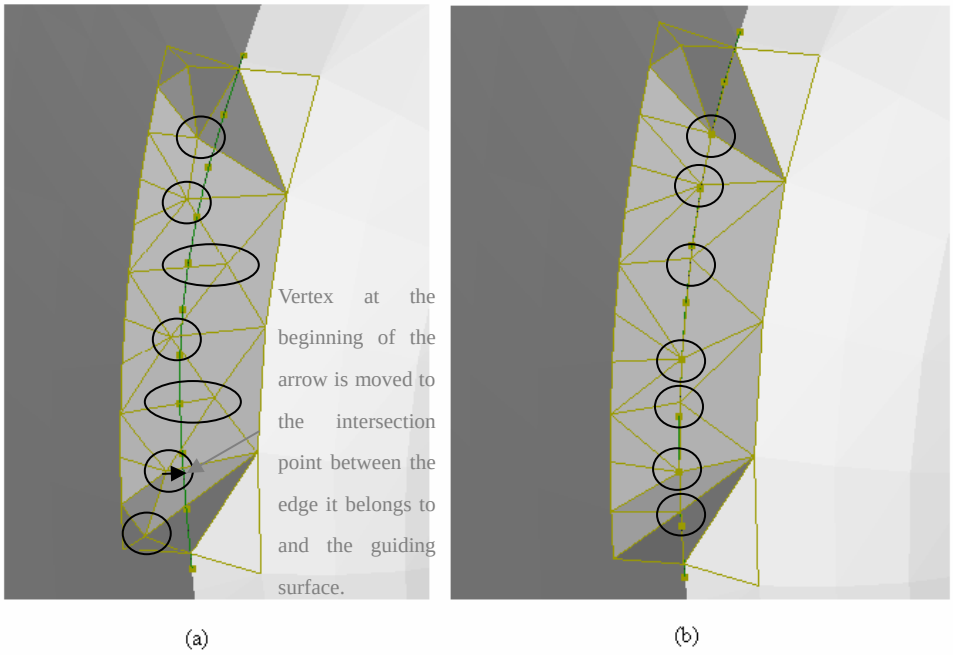


Figure 4.7 shows the result of our vertices snapping algorithm in a mouse model. (a) hole triangles before using our algorithm. (b) hole triangles after using our algorithm.

4.2.3 Surfaces modification using haptic interface device

After aligning triangles through which the guiding surface passes, we push all those triangles down using haptic interface device. The user can first specify how much he/she wants to move these triangles (specify the distance that the triangles will be moved each time when they are in contact with the virtual tool). Then, the user sweeps the tool along the intersection line between the guiding surface and the reconstructed triangles. Hand shaking problem is minimized since the tool is snapped onto the guiding surface. Each time a triangle is touched by the virtual tool, it will be moved down with pre-specified distance until it reaches the Hermite curve or the user stops sweeping the tool. The algorithm used in this process can be described as follows:

Algorithm_VI. Surfaces Modification Using Haptic Interface Device

Input: A hole triangle, proxy point at current time step and proxy point at previous time step. Proxy point is the position of the virtual tool.

Output: A hole triangle with sharp edge.

GET two proxy points, one at the previous time step and another at the current time step, and the triangle that the virtual tool is currently on.

COMPUTE the direction of movement of the tool from the two proxy points
FOR (all vertices of triangle that the virtual tool is currently on)
{
    COMPUTE distance between the vertex and the line representing the tool movement direction.
}
GET a vertice that is closest to the line representing tool movement direction.
COMPUTE the final position of the vertex if it is touched by the virtual tool.
IF (the pre-computed final position passes the Hermite curve)
{
    MOVE the vertex to intersection point between the line connecting the current position and the pre-calculated position of the vertex, and the Hermite curve.
}
ELSE
{
    MOVE the vertex directly to the calculated final position.

}
END

The above algorithm is repeated until the user is satisfied or the vertices of deformable triangle reach the Hermite curve. Next, our final step is to orient hole triangles in order to smooth the surfaces.

4.2.4 Surface smoothing

After the user is satisfied with the sharp edge he/she created using haptic interface device, surface smoothing algorithm is used to smooth other hole triangles such that they approximate the curvature of the hole boundary surfaces.

The concept of our surface smoothing algorithm is that; we want to search along **visited** edges. **Visited** edge refers to an edge that consists of hole boundary points, fixed points or belongs to **fixed** triangle. Points and triangles are treated as **fixed** or unable to be modified when they are located on the hole boundary, on the guiding surface, or when they are already moved. Considering triangle $\Delta P_i P_j P_k$, if point P_k is not fixed and edge e_{ij} is a visited edge. Point P_k can be improved by the following steps.

1. Calculate average of normal vectors of hole boundary triangles surrounding the visited edge.
2. Move the movable point such that normal vector of the triangle, to which the point and the visited edge belong, is the same as the average normal in step 1.

The algorithm is repeated until all the hole triangles are considered (all triangles are treated as **fixed**). Our surface smoothing algorithm can be described in the form of pseudo code as shown by following.

Algorithm_VII. Surface Smoothing

Input: hole triangles

Output: smoothed hole triangles

/* Set value of all related points and edges */

SET hole boundary points and points on the guiding surface to “fixed”.
SET hole boundary edges to “visited”

/* Get the normal information of surrounding triangles */
FOR (all **hole boundary points**)
{
COMPUTE a normal vector N_i at a current hole boundary point P_i by averaging normal vectors of all boundary triangles surrounding the point P_i.
}

REPEAT
{
WHILE (at least one **hole point** left **unconsidered**)
{
/* Check the type of point */
GET a hole point by marching along the visited edge
CLASSIFY_POINT
IF (hole point is type 1)
{
SET this point to considered
MODIFY_POINT1
UPDATE_SURROUNDING_TRIANGLE
}
IF (hole point is type 2)
{
SET this point to considered
MODIFY_POINT2
UPDATE_SURROUNDING_TRIANGLE
}
}
} **UNTIL** (no hole point is modified in above process) /* all triangles are fixed. */
END

Hole points are all the points lying over the hole including hole boundary points. Hole boundary point is subset of hole point.

CLASSIFY_POINT is the process used to classify the point being considered. If the point is already fixed, we check all the triangles surrounding the point whether all vertices of each of them are also fixed. If result of the test is TRUE, we set the corresponding triangle to “fixed” triangle. On the other hand, if the point is not fixed, we then find a number of triangles that satisfies two requirements. First, the triangle must have one visited edge. Second, the triangle must have the point being considered as one of its vertex. If number of triangles satisfying these requirements is one, the point is categorized as type one. Otherwise, the point is categorized as type two.

In MODIFY_POINT1, if the point being considered is point P_k, the visited edge is e_{ij} having end point P_i and P_j and the corresponding triangle is triangle $\Delta P_iP_jP_k$, we calculate the average normal $avgN_{ij}$ of normal vectors N_i and N_j at point P_i and P_j respectively. Then, we rotate the point P_k around the edge e_{ij} such that the final location of point P_k makes normal vector of the triangle $\Delta P_iP_jP_k$ equal to the average normal $avgN_{ij}$. Finally, we set point P_k to fixed point and set the triangle $\Delta P_iP_jP_k$ to fixed triangle.

In MODIFY_POINT2, we follow the same process as in MODIFY_POINT1 except that we have more than one triangle in consideration. Therefore, we calculate the location of point P_k for each triangle and then average them all to get the approximated location for point P_k. Finally, we set point P_k to fixed point and set all corresponding triangles to fixed triangles.

Lastly, in UPDATE_SURROUNDING_TRIANGLE, we check all the triangles surrounding new fixed point P_k. If all vertices of each of the surrounding triangles are also fixed, we set the corresponding triangles to fixed triangles and then recalculate their normal vectors. Also, all edges of the fixed triangles are set to visited edge.

We have spent a few sections describing the haptic-based sharp edge retaining algorithm. In the next section, we will present haptic-based gap bridging algorithm which is intended to solve gaps problem.

4.3 Haptic-Based Gap Bridging Algorithm

In this section, haptic-based gap bridging algorithm is presented. We introduce two methods to repair defective gaps in triangular mesh model. The first method is for the user to use haptic interface device to manually connect or stitch two surfaces together. This method might be slow and tedious but it is quite simple to understand and easy to apply in gap bridging problem. User can also control the shape of the created triangles. Another method is to let user manually create only two triangles enclosing the gap. Then, the gap becomes a hole and can be repaired using simple hole filling algorithm.

4.3.1 Manual Gap Bridging Algorithm

Basic steps of manual gap bridging algorithm are presented in Figure 4.8. We begin with a model with a defective gap separating two surfaces. The surfaces, our input, are assumed to be perfectly registered and ready to be repaired. Readers are referred to Figure 4.9 for visual display of this algorithm.

This algorithm needs to have a pre-processing step. That is, we must let the system search for boundary edges. The hole boundary edges search algorithm was already discussed in the previous chapter. With two surfaces registered, the user uses haptic probe to select a triangle $\Delta P_i P_j P_k$ along the boundary of a surface. Then, the user makes choice of selection by clicking the button on the haptic probe and dragging the virtual tool across an edge e_{ij} toward another surface. Upon reaching the target surface, when the virtual tool is close enough to a boundary point P_t, the boundary point P_t will be highlighted and if the users desire to create a new triangle $\Delta P_i P_j P_t$, he/she can just release the button. These select, drag and release processes are done repeatedly until the defective gap is completely repair.

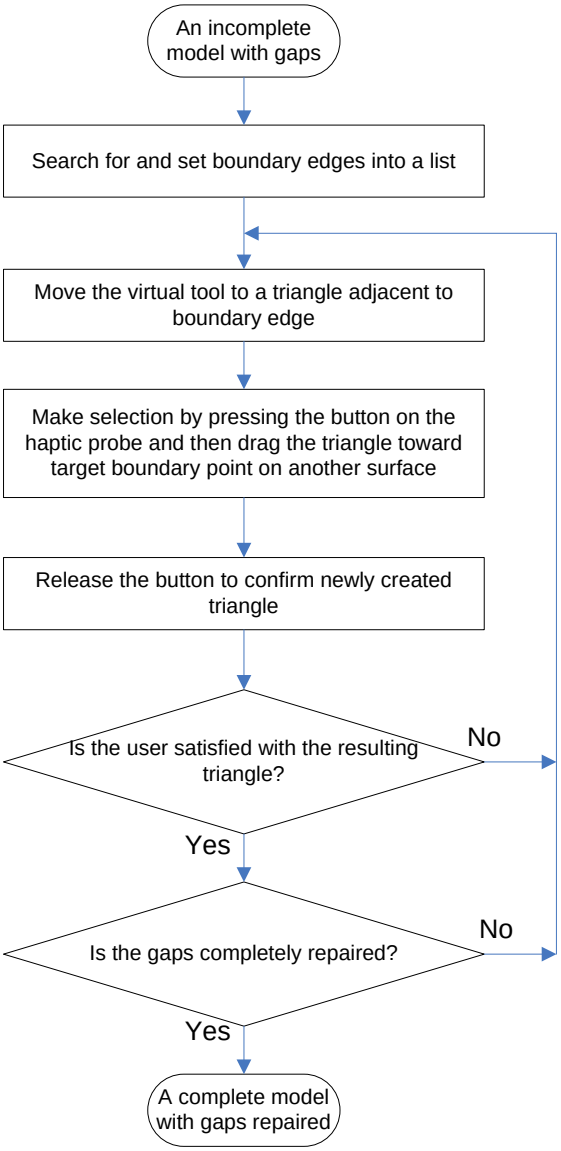


Figure 4.8 **Algorithm VIII: Manual gap bridging algorithm**

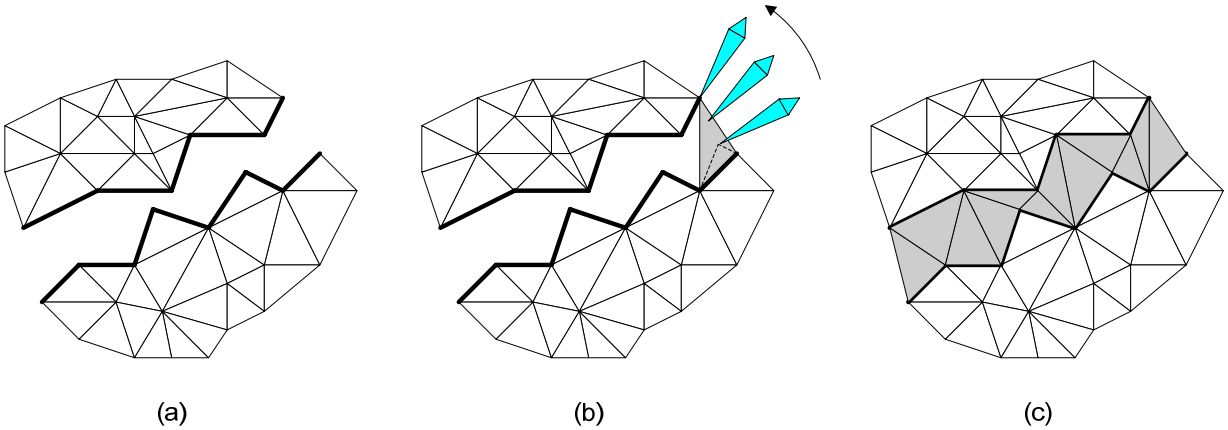


Figure 4.9 Manual Gap Bridging. User uses haptic interface device to manually repair the defective gap. (a) Original surfaces. (b) Haptic interface device is repairing the defective gap. (c) Complete repaired model.

When the user makes selection of a boundary triangle, the system does the following tasks.

```
IF( the button is pressed)
{
    DrawNewTriangle();              /* draw new generate triangle */
    DrawPoints();                   /* draw boundary points */
    GetGroupSnapPoints();           /* Get list of possible snap points */
    GetSnapPoint();                 /* Get a snap point closest to virtual tool */
    IF(number of possible snap points > 0)
    {
        HIGHLIGHT the point
    }
}
END
```

On the other hand, when the user releases the button to confirm new triangle creation, the system does the following tasks.

SET the flag to indicate that the button is not pressed any more
IF(number of possible snap point > 0)
{
    SET new vertex to the closest snap point
    SET new triangle
    CLEAR the list of possible snap points
}

This algorithm is easy to understand and simple to use especially for a gap that has relatively short length. However, it becomes quite tedious if the length of the gap is long. Therefore, in the following section, we introduce another algorithm that might be useful in repairing lengthy gaps.

4.3.2 Semi-automatic Gap Bridging Algorithm

We present a semi-automatic gap bridging algorithm in this section. Basic steps of the algorithm are presented in Figure 4.10. We begin with a model with a defective gap separating two surfaces. The surfaces, as our input, are assumed to be perfectly registered and ready to be repaired. Readers are referred to Figure 4.11 for visual display of this algorithm.

This approach is a modification of manual gap bridging algorithm. Rather than using haptic interface device to connect and create triangles along the gap, the user just needs to create two triangles at both end of the gap to create a closed boundary. Then the gap turns into a hole and can be repaired using hole filling algorithm.

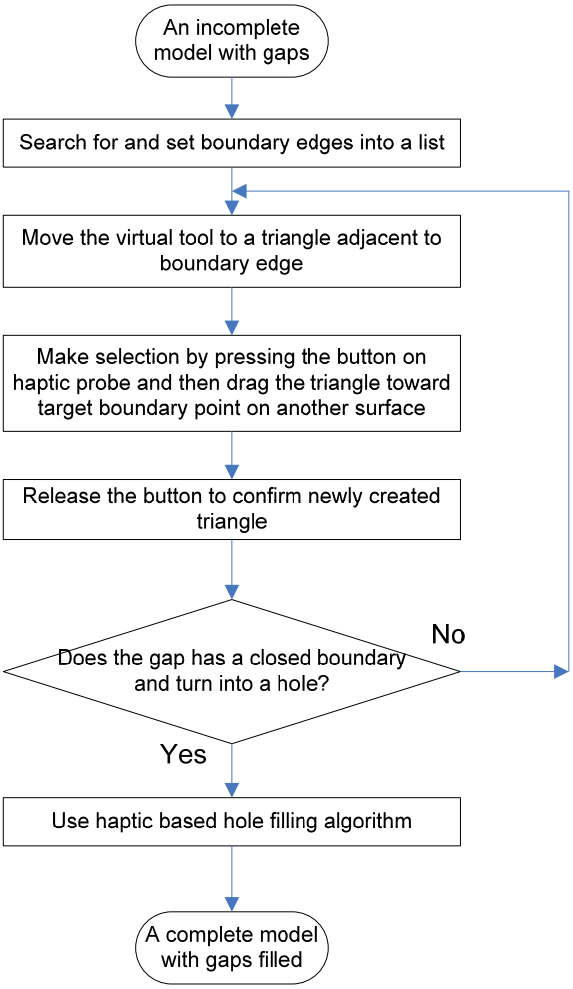


Figure 4.10 **Algorithm IX: Semi-automatic gap bridging algorithm**

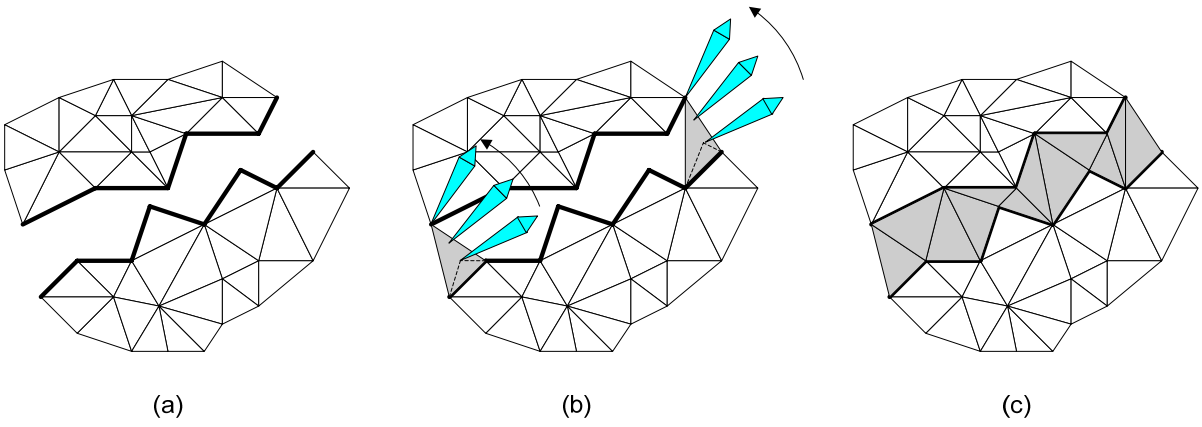


Figure 4.11 Semi-automatic Gap Bridging. (a) Original meshes. (b) Haptic device is used to create two triangles enclosing the defective gap. (c) The gap after being applied with hole filling algorithm.

The semi-automatic gap bridging algorithm eliminates some tedious works (e.g. when using the manual algorithm, the user needs to manually create triangles until the defective gap is completely repaired) and is much faster than manual gap bridging algorithm. However, it is less flexible in the sense that the user cannot control the shape of created triangles since the gap is filled by automatic hole filling algorithm.

4.4 Summary

In this chapter we present a few approaches to repair defective holes with sharp edge retained, and defective gaps in 3D virtual object. Haptic interface device plays an important role in our technique. It is user-friendly and its ability to access and feel the 3D object in virtual space makes it easier to manually create or repair triangular meshes. In the next chapter, we will show some illustrative examples of our haptic-based hole filling, triangle subdivision, surface smoothing and haptic-based sharp edge retaining algorithm.

Chapter 5

Examples and Results

In this chapter, system implementations and illustrative examples of the proposed algorithms are presented. The proposed methods have been implemented on a dual 2.4 GHz CPU workstation using Microsoft® Visual C++ and OpenGL® library at our research lab. The implemented system has been integrated with a PHANTOM Desktop™ Haptic Interface Device from SensAble Technologies and implemented using the OpenHaptic™ toolkit.

5.1 Examples of models with Sharp Edge Resulted From Haptic-Based Hole Filling and Sharp Edge Retaining Algorithm

Figure 5.1 shows the general concept of the proposed haptic-based hole filling and sharp edge retaining algorithm. Having a 3D virtual object model in the form of STL file format as the input, the user can use haptic interface device to touch and feel the surface of the object. Figure 5.2 illustrates the implemented haptic-based system and the user interface. Whenever the user moves the virtual tool into an area close enough to a defective hole, the hole boundary will be highlighted. Then, if the user would like to repair the hole, he/she can initiate haptic-based hole filling algorithm by simply pressing a button on the haptic probe. The selected hole will be automatically repaired using a series of methods which we already described in Chapter 3. The final result of haptic-based hole filling algorithm is the model with a repaired hole. However, the repaired surface might not well approximate the curvature of the boundary surfaces. To improve it, surface smoothing algorithm can be done separately after the hole is completely filled. The optional surface smoothing algorithm will be operated upon the user's call.

To initiate haptic-based sharp edge retaining algorithm, the user is required to select four points along the sharp edge on the hole boundary surface as we described in Chapter 4. These points will be used to create a Hermit curve which approximates sharp edge across the hole area. After that, the algorithm generates a guiding surface such that it passes through the Hermite curve and penetrates through repaired surfaces over the hole. The repaired triangular patches through which the guiding surface penetrates are re-aligned such that their vertices located closest to the guiding surface are snapped onto the guiding surface. This vertices snapping is not applied for the hole boundary points. Then, the user is free to use haptic probe (represented as the virtual tool on the screen) to sweep over the repaired triangular patches along the guiding surface and push them down to the satisfied level. The user can stop sweeping when he/she satisfies with the resulted sharp edge. If the user keeps sweeping on the triangles, the repaired triangular patches will finally reach the Hermite curve approximating the sharp edge and will not be able to move anymore. Surface smoothing algorithm is an optional choice for the user. The user can decide to use the algorithm after the sweeping process if he/she wants to improve other triangles that are not moved during the previous step.

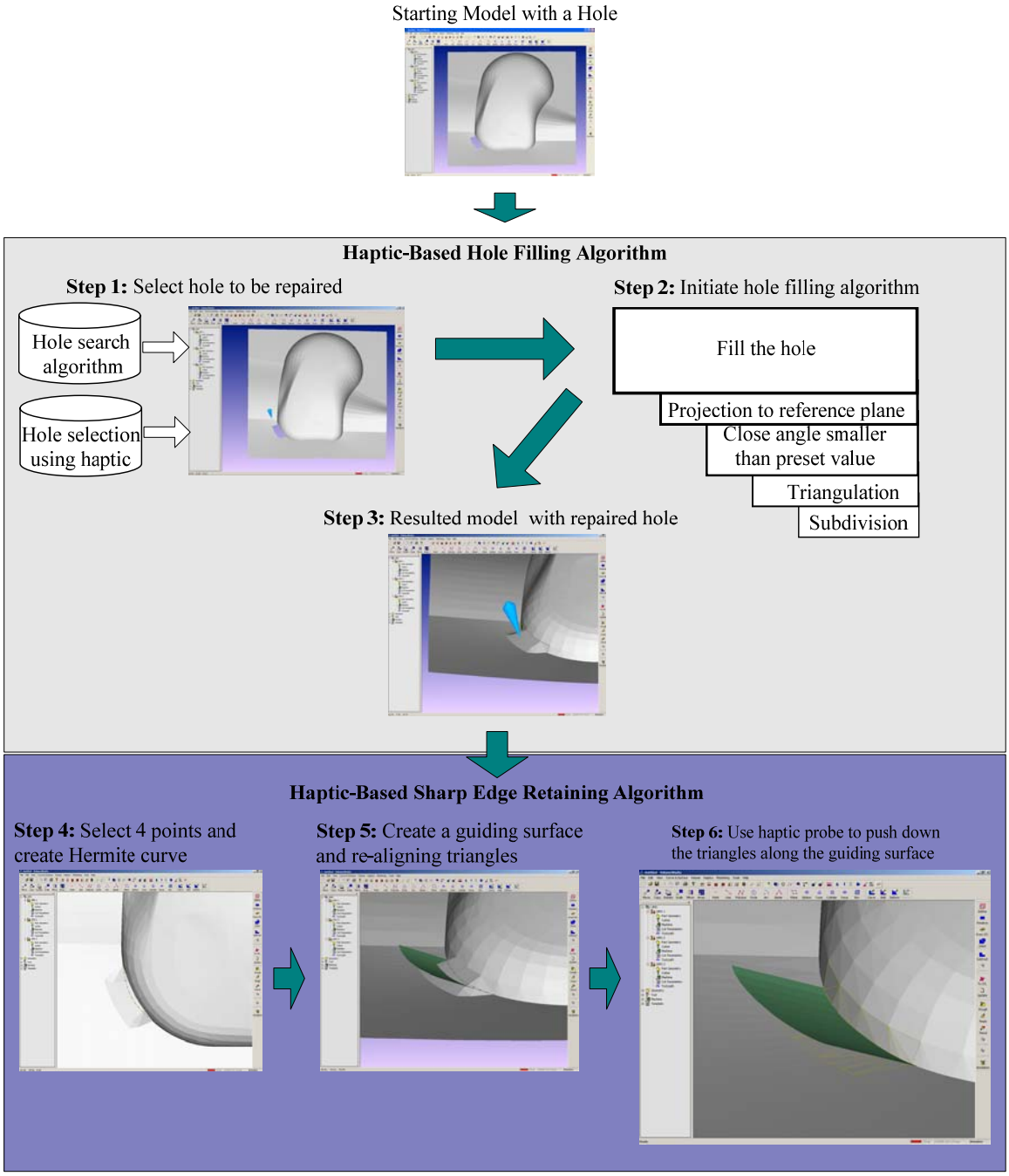


Figure 5.1 General concept of the proposed haptic-based hole filling and sharp edge retaining algorithm.

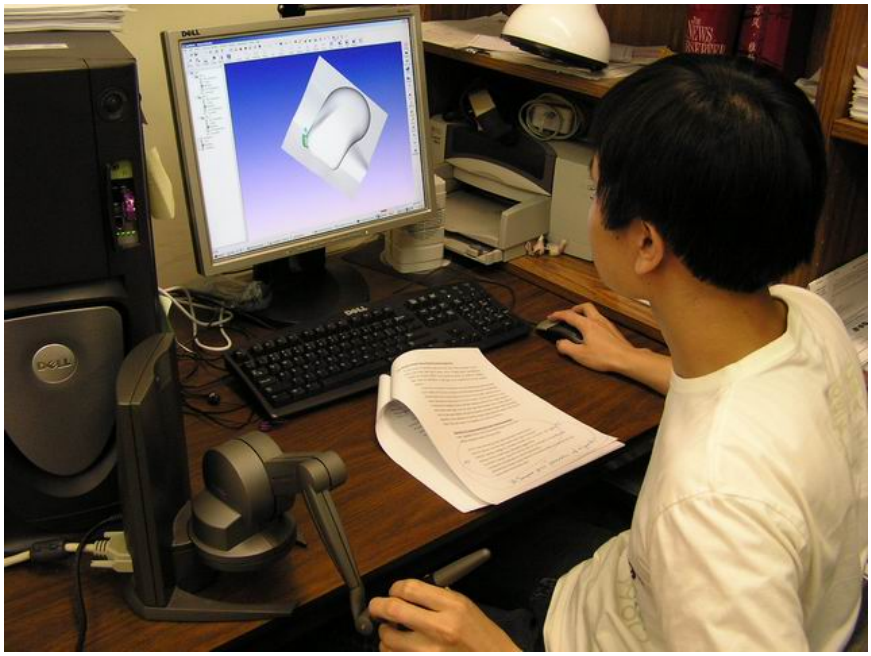


Figure 5.2 Our lab setup haptic interface device

The next few figures show examples of the haptic-based hole filling and haptic-based sharp edge retaining algorithm. Figure 5.3 illustrates our first example, the mouse model. The model has a defective hole at the bottom left corner of the mouse body and exactly on the sharp edge. We used our haptic-based hole filling and sharp edge retaining algorithm to repair and create the sharp edge over the defective hole. Figure 5.4 shows the model after hole-filled and four points are selected for generating a guiding surface. Figure 5.5 illustrates the guiding surface and the repaired surface being pushed down by the haptic interface device. The resulted model after sharp edge retained is shown in Figure 5.6.

The repaired sharp edge illustrated in Figure 5.6 is created by setting the Hermite's slope parameter k_1 and k_2 = 15. The parameter k_1 and k_2 can be thought of as the length of the tangent vectors at point P_2 and P_3 (the two among four selected points that are located closest to the hole) as we described in Chapter 4.

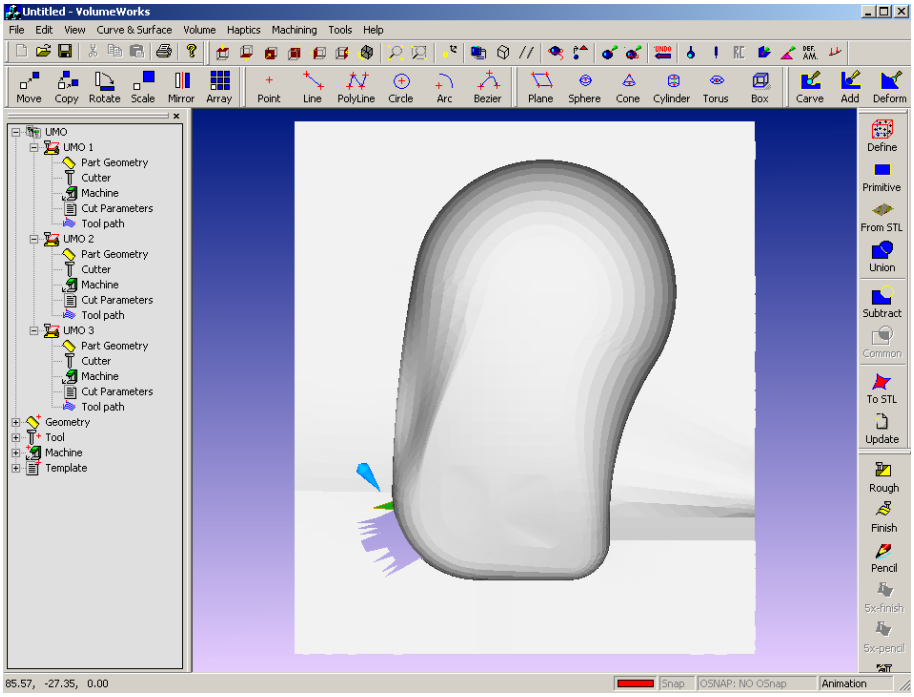


Figure 5.3 A mouse model with a defective hole at the bottom left corner. The blue cursor floating on top of the object is our virtual tool.

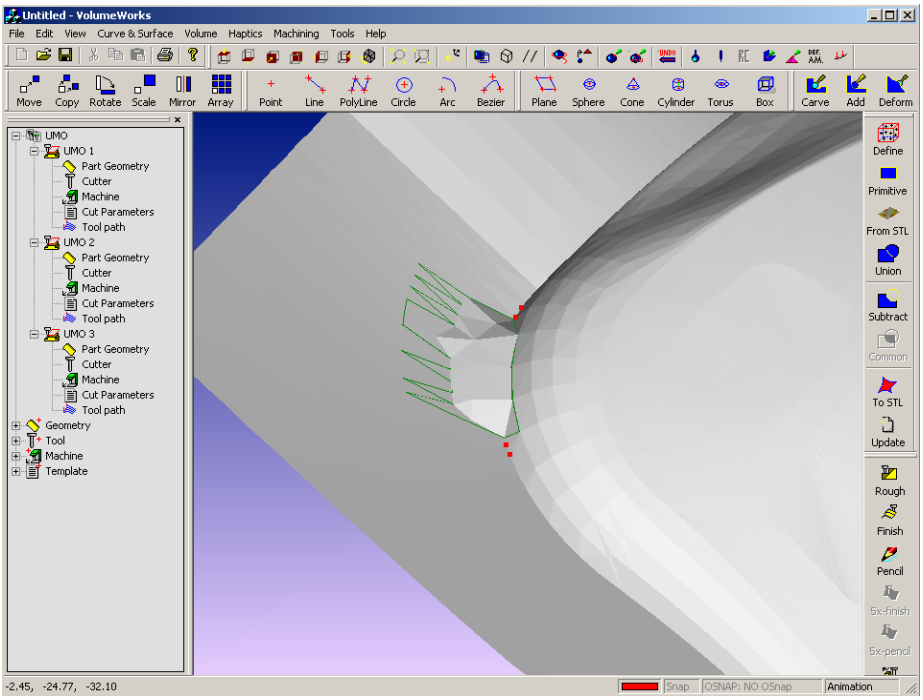


Figure 5.4 The mouse model with a defective hole after hole filled.

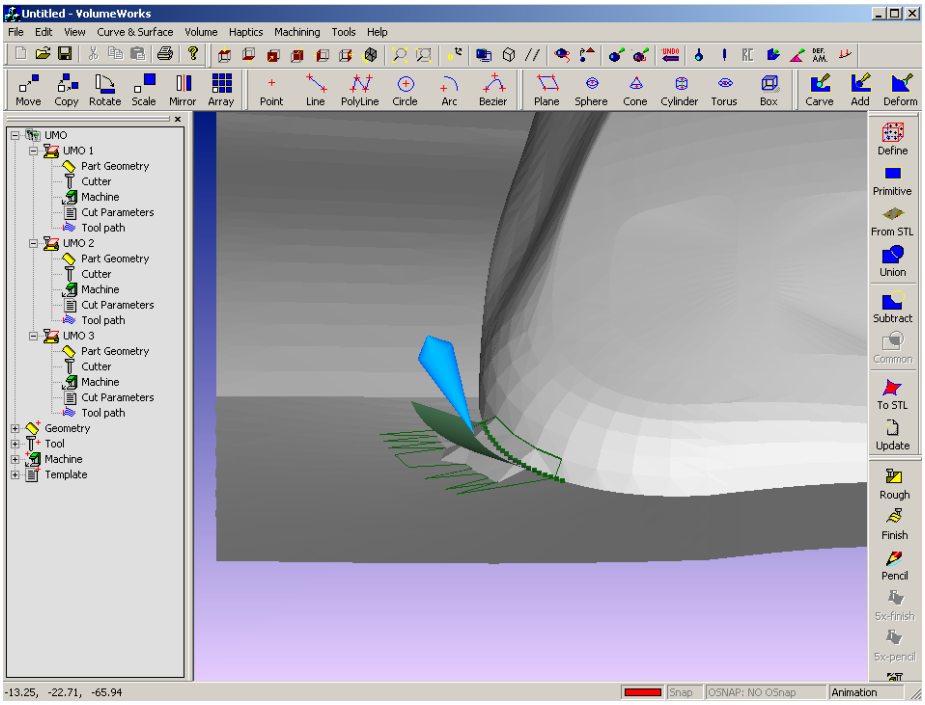


Figure 5.5 The guiding surface and the repaired surface being pushed down by the haptic interface device

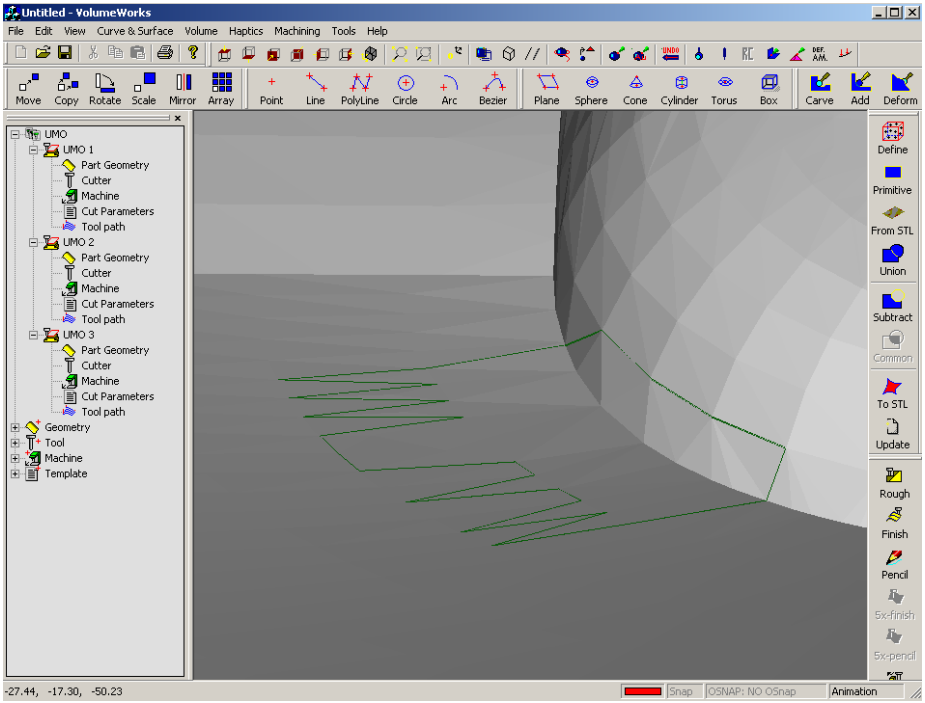


Figure 5.6 The resulted model after sharp edge retained.

From observation on the mouse model, it appears that our algorithm works pretty well. The retained sharp edge looks almost similar to the original one and fits perfectly to the sharp edge outside the hole boundary. However, since k_1 and k_2 parameters used to control the curvature of sharp edge must be determined by the user, failure to obtain best appropriated parameters may create less accurate result.

From this example, we may conclude that, for the geometrical parts that their sharp edges are quite obvious and curvature information is not too hard to obtain, our algorithms could provide satisfied result. The algorithms can repair the non-watertight model and transform it into the watertight one by filling the defective hole and having the sharp edge retained. Major factors that are able to bias the accuracy of the result are positions of four points for creating Hermite curve and determination of the parameter k_1 and k_2.

Another example that we would like to show next is a foot model with a defective hole as illustrated in Figure 5.7. We use our haptic-based hole filling to repair the hole and use haptic-based sharp edge retaining to create the curvature of the surface. The result of our algorithm can be found in Figure 5.8 and the comparison is shown in Figure 5.9.

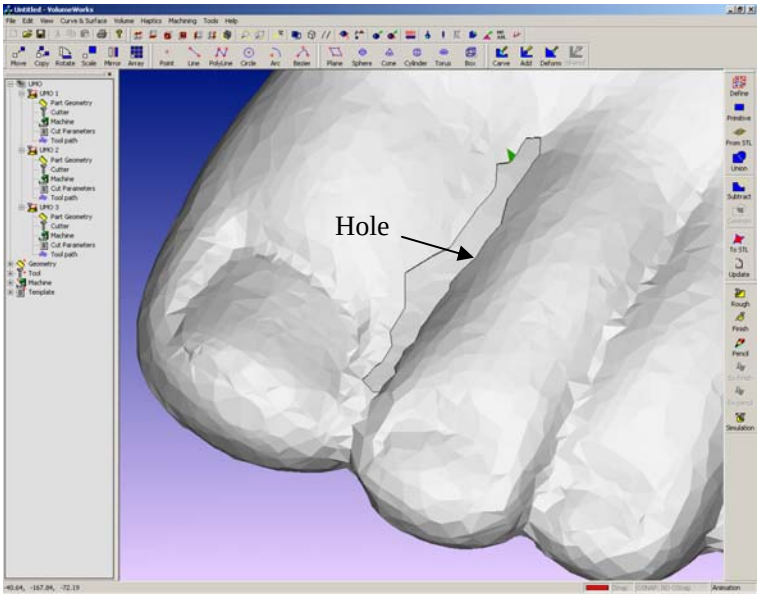


Figure 5.7 A foot model with a defective hole at the toe area.

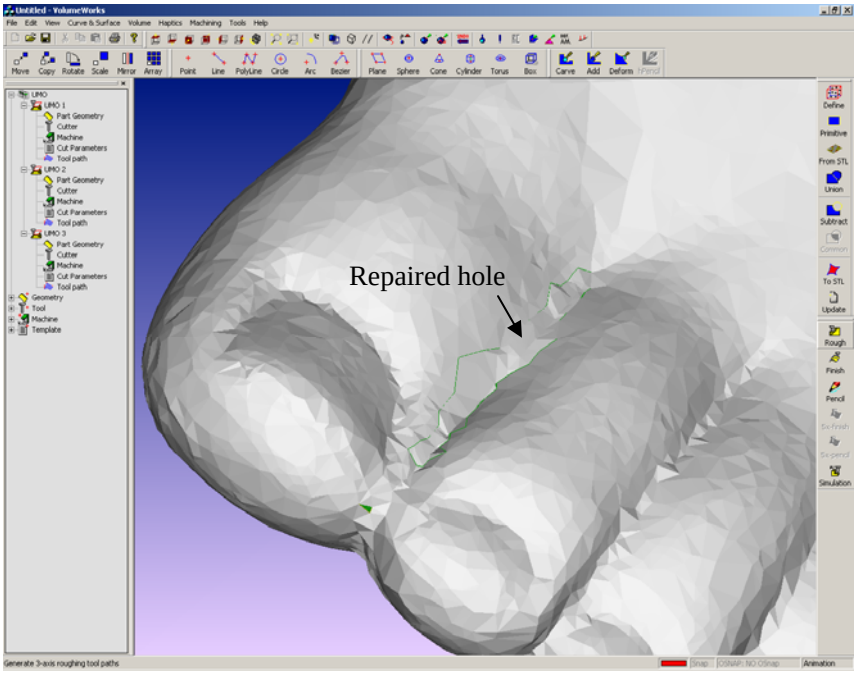
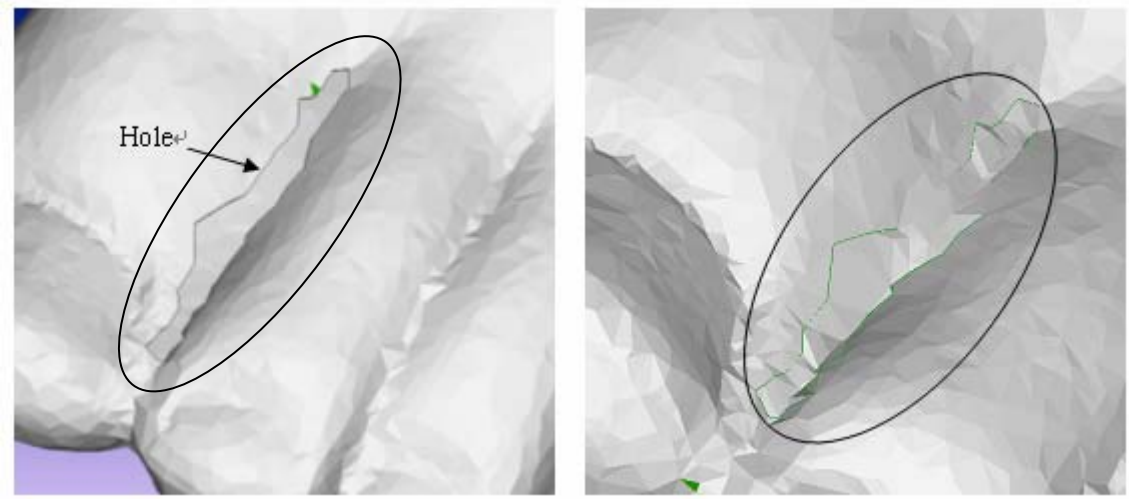


Figure 5.8 A foot model with the defective hole repaired. We use the same setting for parameters k_1 and k_2 as in the mouse model. The green line indicates hole boundary.



(a) (b)

Figure 5.9 The visual comparison between the original and the repaired model. (a) The original surface. (b) The repaired surface from our algorithm.

Based on visual inspection, the result looks fine to us. We can see a valley that represents our reconstructed sharp edge along the toe. Our repaired surface fits well and connects seamlessly to the hole boundary.

However, we can identify some problems that might be worth discussed here. For the free-form model (e.g. the foot model), we found some difficulty in selecting four points on the surface in order to create the hermite curve. If any of these points is selected on a bad-oriented triangular patch, it will bias the hermite curve and possibly create an incorrect curve that does not well approximate the curvature of the boundary surface. A solution to this problem is to let user deform the triangles along the reconstructed sharp edge as much as he/she wants in order to compensate for this inaccuracy. In other words, the deformation process should not stop even when the deformable triangles reach the approximated sharp edge.

In summary, we conclude that our concepts give acceptable results for both the geometrical and free-form parts. Although, there is a difficulty in defining sharp edge as we mention above, a good result can be obtained by carefully selecting the points for constructing Hermite curve. Our algorithms are able to create watertight model with sharp edge retained from a non-watertight one. The concepts can be useful in Computer Aided Design (CAD) and Reverse Engineering (RE) applications.

5.2 Examples of models Resulted from Haptic-Based Gap Bridging Algorithm

Figure 5.10 shows the general concept of the proposed haptic-based gap bridging algorithm. We start with an input model with a defective gap in presence. Using a 3D virtual object model in the form of STL file format as an input, the user can use haptic interface device to touch and feel the surface of the object. To repair a gap, the user has two options to finish the work.

In the first method, the user simply moves the virtual tool to a boundary triangle and “click and drag” the triangle towards a target point that he/she wants to form a new

triangle with. During “click and drag” all the boundary points will be highlighted and thus make it easier for the user to find the appropriated target point. When the virtual tool moves closer to the target point, a triangle is tentatively created and snapped to that point. The user can confirm the existence of new triangle by releasing the button to stop “click and drag” process. These steps can be repeated until the gap is completely repaired.

In the second method, we improve the algorithm so that it works faster. With a defective gap in a model, we can create two new triangles at each ends of the gap. Thus, the gap becomes a hole (since it has a closed boundary) and we can use our hole filling algorithm to handle the remaining work.

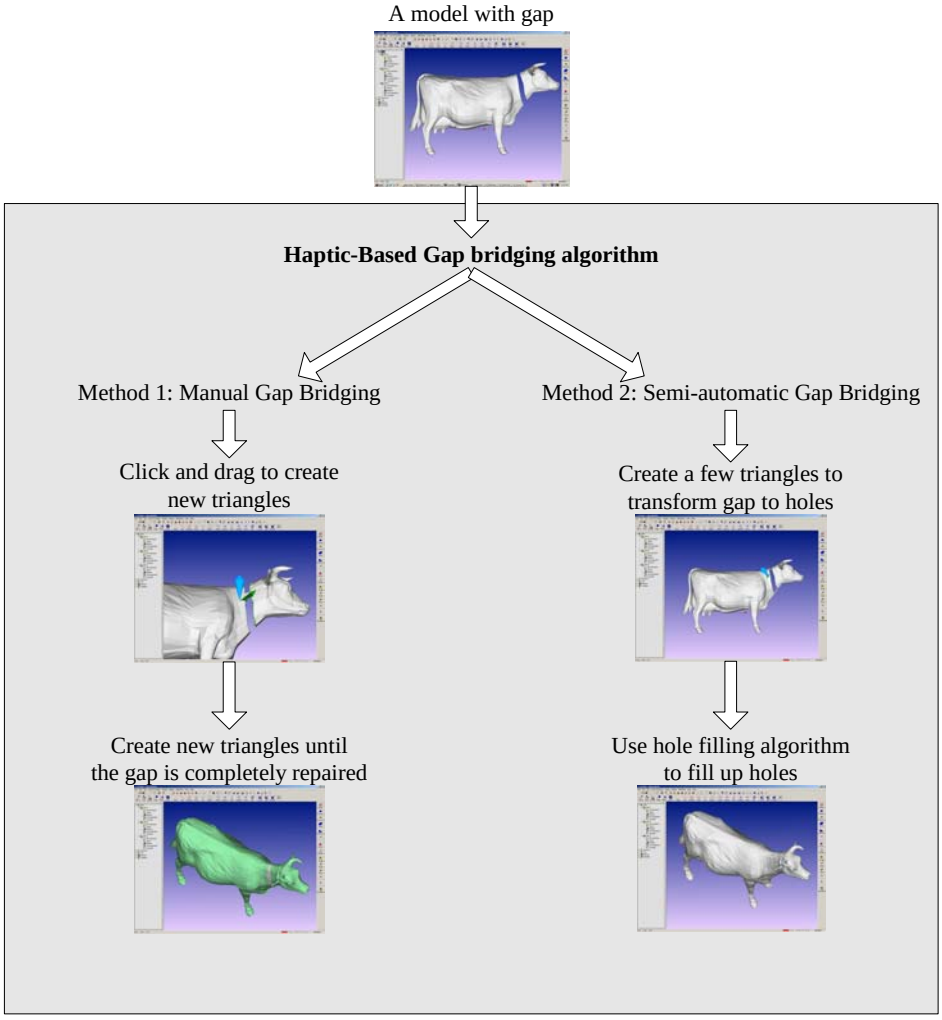


Figure 5.10 The general concept of the proposed haptic-based gap bridging algorithm.

We use a cow model as an example for our haptic-based gap bridging algorithm. The cow model with a defective gap is shown in Figure 5.11. In this model, we assume that the two surfaces are perfectly registered and ready to be repaired.

Using our first method, manual gap bridging, we get the virtual tool close to a boundary triangle and “click and drag” to create a new triangle. We follow the steps described in Chapter 4 and completely repair the gap. The “click and drag” process and our resulted cow model are shown in Figure 5.12.

Our second method is to use the manual gap bridging to create a few new triangles such that the gap becomes holes. Then these holes will be repaired using our haptic-based hole filling algorithm. The result of this method can be found in Figure 5.13.

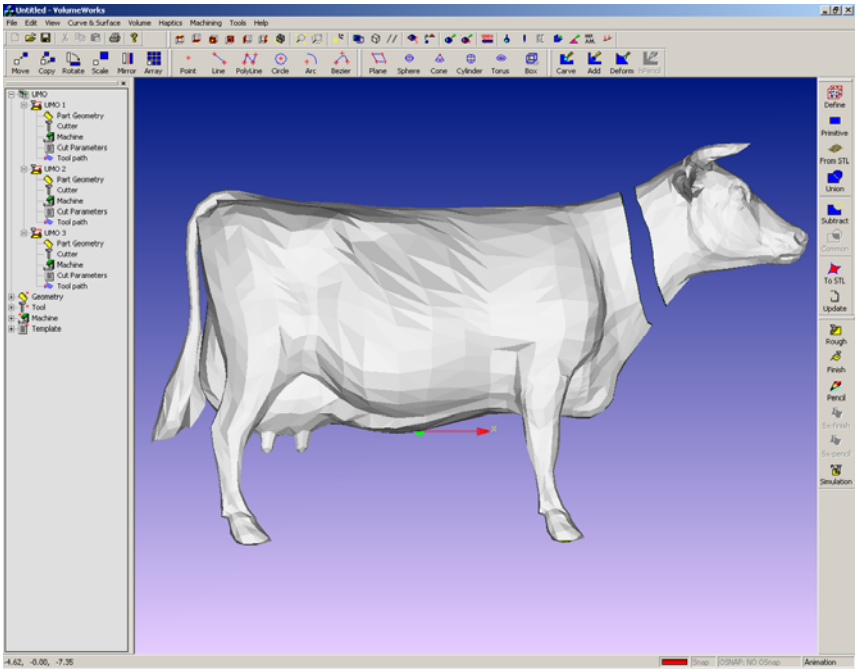
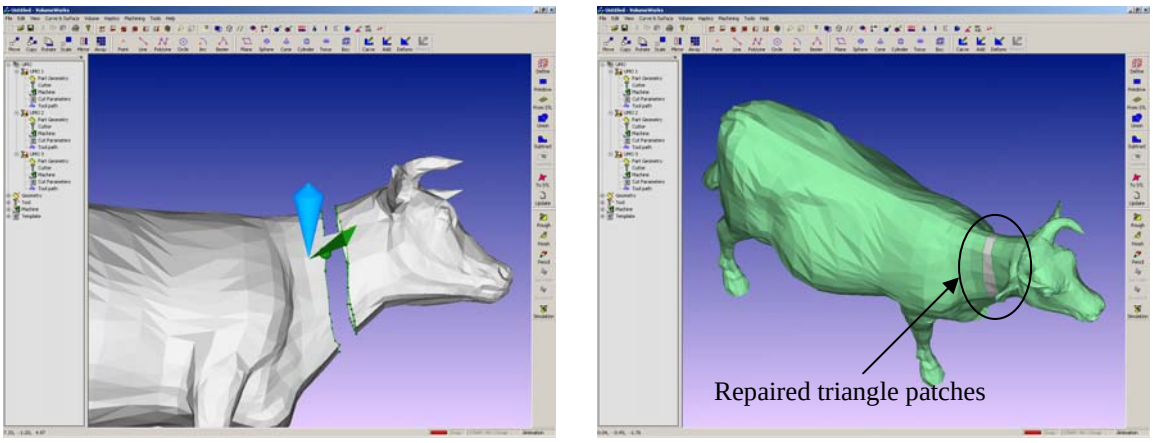
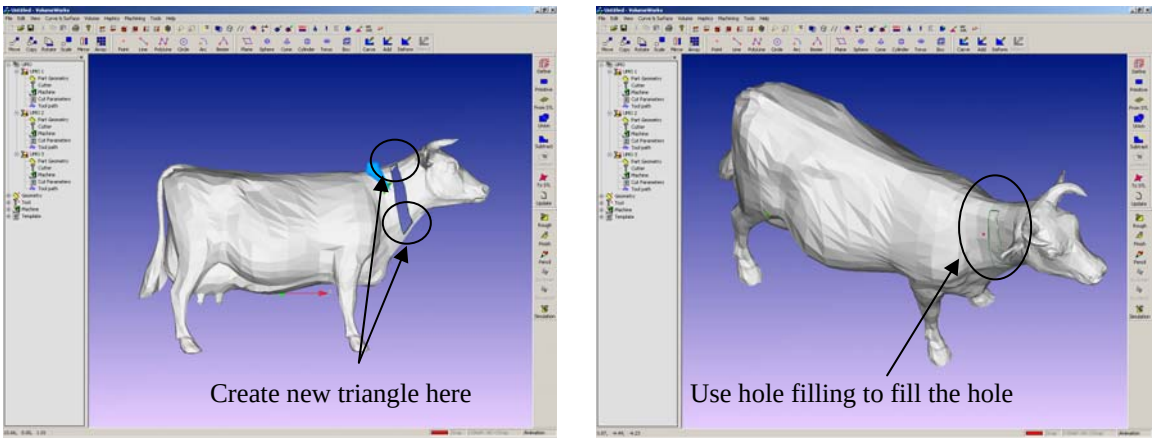


Figure 5.11 A cow model with a defective gap.



(a) (b)

Figure 5.12 Repaired cow model using manual gap bridging. (a) “Click and drag” process. (b) The resulted cow model from our manual gap bridging. The gray strip indicates the repaired triangle patches.



(a) (b)

Figure 5.13 Repaired cow model using semi-automatic gap bridging. (a) We use manual gap bridging to create a few triangles. (b) The resulted cow model from our semi-automatic gap bridging.

Another example for haptic-based gap bridging algorithm is the S-model as shown in Figure 5.14. The model has defective holes and gaps all over its surface. What we want to do is that we will keep the large hole at the middle of the part and seal the defective gaps on the width of the part. This problem cannot be solved by hole filling algorithm since the defective gaps does not have a closed boundary. Hole filling algorithm will fill the middle hole rather seal the gap on the width of the part. Figure 5.15 illustrates the part after modified.

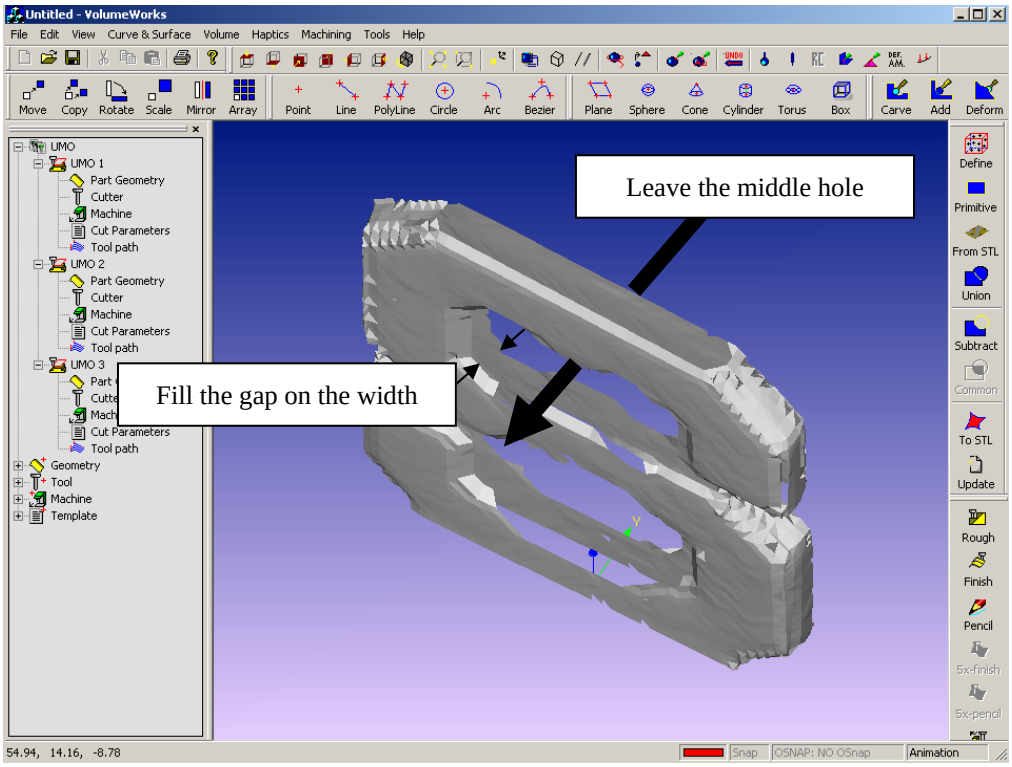


Figure 5.14 The S-model with defective holes and gaps, originally reconstructed from scanned data

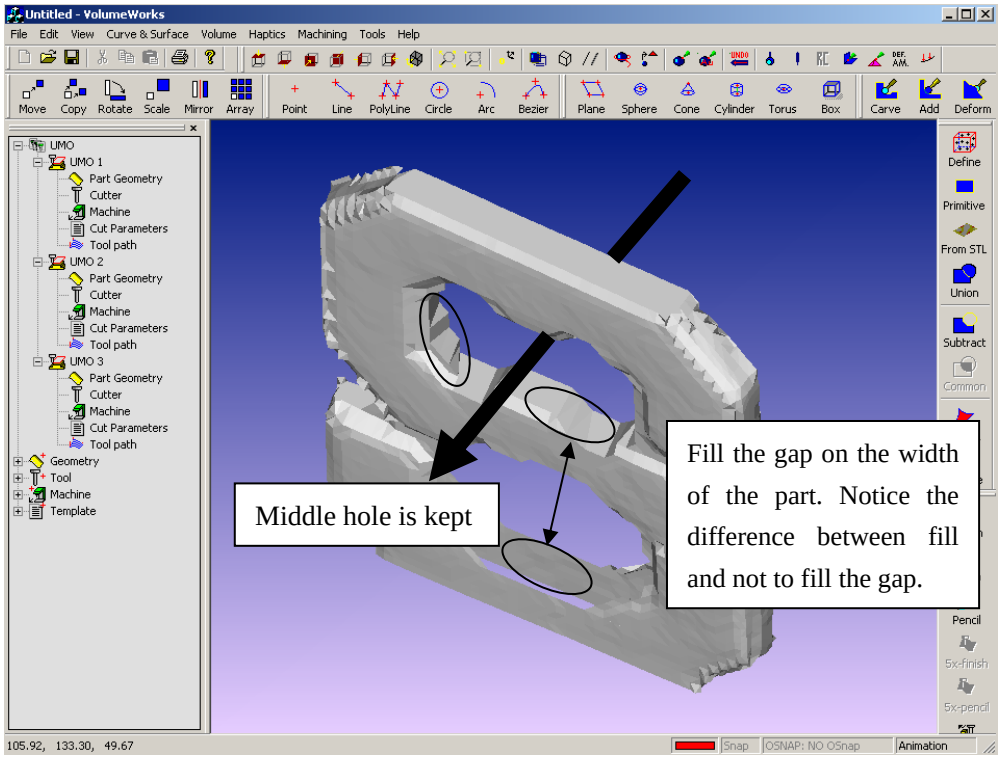


Figure 5.15 The S model after gap filled using the haptic-based gap bridging technique.

From the example shown above, our haptic-based gap bridging algorithm provides satisfied result. The repaired surface fits well with its boundary and the model transforms into a watertight model. However, we found some difficulties in our first method, the manual gap bridging. If the size of the model is relatively small or the user cannot hold his/her hand still, it will not be easy to create new triangles since the virtual tool could move very fast so that the “click and drag” cannot be easily obtained. Also, it is quite tedious to repair the defective gap by creating triangles one by one. Unless the shape of repaired surface is highly important, this method may not be preferable since the user needs to spend some time with it. Our second method is better in the sense that it works faster, but it depends on the shape of gap as well. If the gap is pretty wide, subdivision and surface smoothing should be done after the process of creating triangles in order to obtain better surfaces. Therefore, the appearance of the repaired surface relies heavily on the surface smoothing algorithm.

5.3 Summary

In this chapter, illustrative examples of the proposed algorithms have been presented. The results show that our haptic-based hole filling algorithm can support the user or designer to easily and selectively repair the defective hole in need. Haptic-based sharp edge retaining algorithm can be used to create sharp edge over the repaired surface such that the surface fits well with its boundary. Finally the haptic-based gap bridging algorithm can be a choice for the user in repairing gaps in 3D virtual model. These algorithms could be used in Computer Aided Design (CAD) and Reverse Engineering (RE).

Chapter 6

Conclusions and Future Research

6.1 Conclusions

In this thesis, we have presented the computational techniques and algorithms that can be used in repairing defective holes and gaps in non-watertight polyhedron models. The constrained intrinsic property driven algorithm transforms point cloud data into a surface model with elimination of abnormal triangular patches. Our haptic-based hole filling and sharp edge retaining algorithms provide a new opportunity to repair and retain sharp edge in free-formed parts, while they can be used in solving hole and sharp edge problems in geometrical parts as well. Haptic-based gap bridging algorithm is a conceptual technique that can be used to repair defective gaps. The presented techniques have been implemented and integrated with a haptic force feedback device at our lab. It is very intuitive and can be done in real time. These algorithms provide a feasible tool to help and to facilitate the repair of non-watertight models. The contributions in this paper can be summarized as follows:

- Haptic-Based Hole Filling algorithm provides a choice for the user to selectively fill defective holes or gaps in the incomplete object models. The user might decide to leave some holes unfilled or completely repair defective holes in the model. We integrate a 6-DOF (Degree Of Freedom) PHANTOM Desktop™ Haptic Interface Device from SensAble Technologies into our system. With the haptic force feedback device, the user can feel the surface and access to the hole that is normally difficult to reach by regular 2D mouse.
- Haptic-Based Sharp Edge Retaining algorithm is developed such that it can create sharp edge on the repaired surface in both geometrical and free form model. The algorithm is flexible enough to let user decide the curvature of the sharp edge by adjusting the parameters for Hermite curve. Although depth of the sharp edge is defined by the Hermite curve, the user can adjust degree of depth using the

intergrated 6-DOF (Degree Of Freedom) Haptic Interface Device.

- Haptic-Based Gap Bridging Algorithm shows another application of Haptic Interface Device. This algorithm can be used as a tool to manually or semi-automatically repair gaps in non-watertight model. It is also intuitive and can be operated in real time environment.

The computer implementation and integration with the haptic device demonstrates that the proposed techniques can repair incomplete or defective triangulated facet models and recover sharp edge based on user's setting parameters. Force and torque feed back device provides more flexibility for users to selectively repair only the defective holes and gaps. More importantly, it can be used to gather information from a vague surface object which cannot be done by other automatic computational techniques.

6.2 Future Research

There are some improvements that could be applied to the proposed algorithms so that they can operate on various types of models and provide more ease of use to the user. The future research can be proposed as follows.

- Implement line snapping function such that the sharp edge retaining algorithm can be used to create sharp edge onto the convex surface.
- Real time adjustment of Hermite curve parameters k_1 and k_2 would make the program easier to use and reduce the time user needs to take to obtain an appropriated values.
- In the free form model, depth of sharp edge can be more flexible if it is not limited to the Hermite curve. The program can be modified such that the user can increase depth of sharp edge beyond the pre-approximated one.
- Smoothness of the constructed surface could be improved by using better surface smoothing algorithm.

References

[Adamy 2002] Adamy U, Giesen J, John M. Surface reconstruction using umbrella filters. Computational Geometry 2002; (21): 63-86.

[Amenta 1998] Amenta N, Bern M, Kamvysselis M. A new Voronoi-based surface reconstruction algorithm. Computer Graphics (Proceeding of SIGGRAPH'98) 1998; July:19-24.

[Amenta 1999] Amenta N, Bern M. Surface reconstruction by Voronoi filtering. Discrete and Computational Geometry 1999; 22(4): 481-504.

[Attene 2003] Attene M., Falcidieno B, Rossignac J and Spagnuolo M. Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces. Proceedings of 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing. pp 62-69. 2003.

[Bechmann 2006] Bechmann, D. Modélisation Géométrique course note. 2006. https://dpt-info.u-strasbg.fr/~bechmann/Subdivisions.pdf

[Bernadini 1997] Bernardini F, Bajaj C, Chen J, Schikore D. Triangulation-based object reconstruction methods. Proceedings of SCG'97 (ACM Symposium on Computational Geometri); 1997 481-484.

[Bernaradini 1999] Bernaradini F., Bajaj CL., Chen J. and Schikore DR. Automatic reconstruction of 3D CAD models from digital scans. International journal on computer geom. and applications., vol. 9, nos. 4-5, Aug & Oct 1999, pp. 327-369.

[Bernadini 1999] Bernardini F., Mittleman J., Rushmeier H., Silva C. and Taubin G. The ball-pivoting algorithm for surface reconstruction. IEEE Trans Visualization Computer Graphics 1999; 5(4): pp. 349-359.

[Boissonat 2000] Boissonat J-D, Cazals F. Smooth surface reconstruction via natural neighbor interpolation of distance functions. Proceedings of SCG'00 (ACM Symposium on Computational Geometry); 2000, 223-232.

[Boonma 2006] Boonma A. and Lee Y-S. Haptic-Based Sharp Edge Retaining and Gap Bridging Algorithms for Computer Aided Design (CAD) and Reverse Engineering (RE). Working paper. 2006.

[Chen 2005] Chen C-Y. Cheng K-Y., Liao H.Y.M. A Sharpness Dependent Approach to 3D Polygon Mesh Hole Filling. Proceedings of EuroGraphics 2005, Short Presentations, 13-16, 2005.

[Chua 2003] Chua C.K., Leong K.F. and Lim C.S. Rapid prototyping: principles and applications. 2nd edition, 2003. World Scientific Publishing. ISBN 981-238-117-1

[Chui 2000] Chui C.K. and Lai M-J. Filling polygonal holes using C^1 cubic triangular spline patches. Computer Aided Geometric Design, Volume 17, Issue 4, April 2000, Pages 297-307.

[Curless 1996] Curless B, Levoy M. A volumetric method for building complex models from range images. Computer Graphics (Proceeding of SIGGRAPH) 1996; 303-12.

[Fleishman 2005] Fleishman S., Cohen-Or D. and Silva CT. Robust Moving Least-squares Fitting with Sharp Features. Proceedings of ACM SIGGRAPH 2005. ACM Transactions on Graphics (TOG) vol 24, issue 3, pp. 544-552 (July 2005).

[Foley 1996] Foley J.D., Dam A.V., Feiner S.K., Hughes J.F. Computer Graphics: Principles and Practice. Second edition in C, 1996. Addison-Wesley Publishing Company. ISBN 0-201-84840-6

[Guo 1997] Guo B., Menon J, Willette B. Surface reconstruction using alpha shapes. Computer Graphics Forum 1997; 16(4):177-90.

[Ho 2004] Ho C-C., Lu Y-H., Lin H-T, Guan S-H., Cho S-Y., Liang R-H., Chen B-Y. and Ouhyoung M., "Feature Refinement Strategy for Extended Marching Cubes: Handling on Dynamic Nature of Real-time Sculpting Application," in Proceedings of IEEE 2004 International Conference on Multimedia and Expo, ICME 2004, Taipei, Taiwan.

[Huang 2002] Huang J. and Menq CH. Combinatorial Manifold mesh reconstruction and optimization from unorganized points with arbitrary topology. Computer Aided Design 34 (2002) pp. 149-165.

[Jun 2005] Jun Y. A piecewise hole filling algorithm in reverse engineering. Computer Aided Design 37 (2005) pp. 263-270.

[Karbacher 2001] Karbacher S., Seeger S. and Hausler G. Refining triangle meshes by non-linear subdivision. 3dim, p. 270, Third International Conference on 3-D Digital Imaging and Modeling (3DIM '01), 2001.

[Kirkpatrick 1985] Kirkpatrick DG, Radke JD. A frame work for computational morphology. Computational Geometry 1985; 217-48.

[Lai-Yuen 2005] Lai-Yuen S.K., Lee Y-S., Computer-Aided Molecular Design (CAMD) with Force-Torque Feedback, Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05), pp 199-204.

[Lai-Yuen 2006] Lai-Yuen S.K., Lee Y-S., Energy-Field Optimization and Haptic-Based Molecular Docking and Assembly Search System for Computer-Aided Molecular Design (CAMD), Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS's 06) , p. 34, 2006.

[Liepa 2003] Liepa P. Filling Holes in Meshes. Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing. ACM International Conference Proceeding Series; Vol. 43, Pages: 200 – 205.

[Lin 2004] Lin HW, Tai CL and Wang GJ. A mesh reconstruction algorithm driven by an intrinsic property of a point cloud. Computer Aided Design 36 (2004) pp. 1-9.

[Lu 2002] Lu Y-H., Wang W-T., Liang R-H., Ouhyoung M., Virtual Sculptor: A Feature Preserving Haptic Modeling System, Proc. of ACM International Workshop on Immersive Telepresence (ITP2002), Juan Les Pins, France, Dec. 2002.

[Ma 2005] Ma W. Subdivision surface for CAD-an overview. Computer-Aided Design Volume 37, Issue 7 , June 2005, Pages 693-709.

[Mencl 1998] Mencl R. and Muller H. Interpolation and approximation of surfaces from three-dimensional scatterd data points. In proceeding of Eurographics'98. Eurographics, 1998, State of the Art Reports.

[Otaduy 2005] Otaduy MA. and Lin MC. Introduction to Haptic Rendering. Supplementary Course Notes. Recent Advance in Haptic Rendering and Applications. SIGGRAPH 2005.

[Pfeifie 1996] Pfeifie, R. and Seidel, H.P., Triangular B-Splines for Blending and Filling of Polyg- onal Holes, Proceedings of Graphics Interface '96, pp. 186-193 (1996).

[Ren 2006] Ren Y., Lai-Yuen S. K., Lee Y-S. Virtual prototyping and manufacturing planning by using tri-dexel models and haptic force feedback. Virtual and Physical Prototyping, Taylor & Francis, Volume 1, Number 1 / March 2006, pp 3 - 18

[Salisbury 1995] Salisbury, K., Brock, D., Massie, T., Swarup, N., and Zilles, C. 1995. Haptic rendering: Programming touch interaction with virtual objects. In 1995 Symposium on Interactive 3D Graphics, P. Hanrahan and J. Winget, Eds., ACM SIGGRAPH, 123–130. ISBN 0-89791-736-7.

[Salisbury 2004] Salisbury K., Conti F., Barbagli F., Haptic Rendering: Introductory Concepts, IEEE Computer Graphics and Applications, vol. 24, no. 2, pp. 24-32, Mar/Apr, 2004.

[Tekumalla 2004] Tekumalla LS. And Cohen E. A hole-filling algorithm for triangular meshes. UUCS-04-019, 2004 Technical Report, School of Computing, University of Utah.

[Varady 1997] Varady T, Martin R.R. and Coxt J. Reverse engineering of geometric models – an introduction. Computer-Aided Design, Vol. 29, No. 4, pp 255-268, 1997.

[Veltkamp 1995] Veltkamp RC. Boundaries through scattered points of unknown density. Graphical Models Image Process 1995; 57(6): 441-52.

[Wang 2003] Wang J., Oliveira M.M., "A Hole-Filling Strategy for Reconstruction of Smooth Surfaces in Range Images," sibgrapi, p. 11, XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'03), 2003.

[Yau 2003] Yau H-T., Kuo C-C. and Yeh C-H. Extensino of surface reconstruction algorithm to the global stitching and repairing of STL models. Computer-Aided Design, Volume 35, Issue 5, pp 477-486, 2003.

[Zhu 2004] Zhu W., Lee Y-S. Five-axis Pencil-Cut Machining Planning and Virtual Prototyping with a 5-DOF Haptic Interface, Computer-Aided Design, Vol. 36, Issue 13, 2004, pages 1295-1307.

[Zhu 2004] Weihang Zhu, Yuan-Shin Lee, Dexel-Based Force-Torque Rendering and Volume Updating for 5-DOF Haptic Product Prototyping and Virtual Sculpting, Computer in Industry, Vol. 55, Issue 2, 2004, pages 125-145

[Zhu 2005] Weihang Zhu, Yuan-Shin Lee, A Marching Algorithm of Constructing Polyhedral Models from Dexel Models for Haptic Virtual Sculpting, Robotics and Computer-Integrated Manufacturing, Vol 21, Issue 1, Page 19-36, 2005