Thi My Hanh Ngo
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DéPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

User-guide Features Sensitive Hole Filling for 3D Meshes
TITRE DE LA THÈSE / TITLE OF THESIS

A. Felty
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

J. Lang

P. Morbin

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies
User-Guided Feature Sensitive Hole Filling
For 3D Meshes

Ngo Thi My Hanh

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

©Ngo Thi My Hanh, Ottawa, Canada, 2009
NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L’auteur conserve la propriété du droit d’auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.
Abstract

The 3D models obtained from the 3D scanners are often noisy and have flaws such as gaps, holes, self-intersecting polygons, etc. These defects make them undesirable for various applications. Thus many post processing techniques are needed to apply to improve the quality of the 3D models so that they can be usable in 3D applications. Hole-filling is an important task among them. Most hole filling approaches use techniques to fill up a hole and to smooth the surface without recovering fine features of the original 3D model. This thesis proposes a methodology which can fill holes in a 3D mesh and at the same time preserve sharp features of the geometry of the original model. The main idea is that we reconstruct possible feature curves in the missing parts of the given mesh before filling the hole with smoothing surface. The feature curves in the missing part are reconstructed by extending salient features of the existing parts. Then the hole is partitioned into several smaller and more planar sub-holes divided by the feature curves. By this way, the sub-holes make the hole filling process be more efficient and accurate. User intervention is needed to design fine features or to guide feature curve reconstruction wherever ambiguity exists or results are unsatisfactory. The ambiguity comes from multiple choices to build feature curves out of crest lines of the existing 3D mesh.

Our hole filling techniques is different from other existing techniques as features are taken as the first subject to reconstruct, which eventually drive the feature-definite surface filling process.
Acknowledgements

Numerous people deserve the recognition for their parts in my Master’s study. First of all, I would like to express special thanks to my thesis supervisor, Prof. Won-Sook Lee, for her guidance, support and encouragement to complete my Master’s study. I extend my thanks to my research colleagues at the DISCOVER LAB and the Computer Graphics Applications LAB for their cooperation and discussion during this work, my colleagues at the Institute of Information Technology – Vietnamese Academy of Science and Technology for their encouragement and support. Vietnamese Government and the University of Ottawa had funded my Master’s study, for which I am also very grateful and appreciative. Special thanks go to my family, my dear friends, Tu Nguyen Van, Behnoosh Hariri, Mansee Mongia, Mauricio Vines, David Orellana, Vienna Dang, Andrew Soon, Qian Chen for their deeply understanding, sharing, encouragement and support.
# Table of Contents

Abstract...................................................................................................................................... i

Acknowledgements .................................................................................................................... ii

Table of Contents ......................................................................................................................... iii

List of Figures.............................................................................................................................. vi

Glossary of Terms ....................................................................................................................... x

Chapter 1  Introduction.................................................................................................................. 1

1.1 Motivation.............................................................................................................................. 1

1.2 Problem Statement................................................................................................................ 2

1.3 Proposed Solution................................................................................................................... 3

1.4 Document Organization and System Overview ..................................................................... 4

Chapter 2  Literature Review ....................................................................................................... 6

2.1 Hole filling techniques.......................................................................................................... 6

2.1.1 Volumetric Approaches................................................................................................. 7

2.1.2 Mesh-based Approaches............................................................................................... 9

2.2 Feature recovering and edge preserving hole-filling .......................................................... 12

2.3 Mesh generation..................................................................................................................... 16

2.3.1 Structured meshes and unstructured meshes .............................................................. 17

2.3.2 Delaunay Mesh Generation ........................................................................................ 18

2.3.2.1 Delaunay triangulation............................................................................................ 18

2.3.2.2 Point Insertion....................................................................................................... 21
2.3.2.3 Boundary Constrained Triangulation ........................................22
2.3.3 Advancing front methods ..........................................................22
2.3.4 Quadtree and Octree methods ......................................................25
2.4 Mesh refinement techniques ..........................................................26
2.4.1 Triangle/Tetrahedral Refinement ..................................................26
   2.4.1.1 Edge Bisection ..................................................................26
   2.4.1.2 Edge relaxation ..................................................................28
2.4.2 Quad/Hex Refinement .................................................................29

Chapter 3  User-guided Feature Sensitive Hole Filling ..........................30

3.1 Overview of hole filling ..................................................................30
   3.1.1 Terminologies ........................................................................30
   3.1.2 System overview ....................................................................32
3.2 Crest lines detection ......................................................................33
   3.2.1 Differential geometry background ..........................................33
   3.2.2 Tracing and thresholding crest lines .......................................35
      3.2.2.1 Detection of crest points ................................................36
      3.2.2.2 Connecting crest points ...............................................36
      3.2.2.3 Thresholding ................................................................37
3.3 Hole identification .........................................................................38
3.4 Hole partitioning ...........................................................................39
   3.4.1 Feature points detection ..........................................................39
   3.4.2 User intervention ....................................................................40
      3.4.2.1 The range of movement of a crest point during the user correction.40
3.4.2.2 Calculate the initial virtual point at each corner hole ......................41
3.4.3 Points interpolation using Catmull-Rom spline ...............................42
3.4.4 Subdivision of a hole by hole boundary tracing ..............................46
3.5 Filling a hole through triangulation ..............................................47
  3.5.1 Projection plane calculation .....................................................47
    3.5.1.1 Defining the projecting operator .....................................48
    3.5.1.2 Calculating the rotation matrix ......................................49
  3.5.2 Triangulating the projected hole boundaries ..................................51
3.6 The summary of hole-filling algorithm ..........................................51

Chapter 4 Results and Validation ......................................................52
  4.1 Experiment results .........................................................................52
  4.2 Comparison to other algorithms ....................................................57

Chapter 5 Conclusions ..........................................................................60
  5.1 Contributions ..................................................................................60
  5.2 Discussion .......................................................................................61
  5.3 Future Research .............................................................................63

References .............................................................................................65
List of Figures

Figure 1: The framework of our user-guided feature sensitive hole filling system ....................... 5

Figure 2: An octree decomposition of the bunny model (reproduced from [PR05]) ............... 8

Figure 3: Projection of polygonal holes on a plane. (a) Simple hole; (b) Complex hole
(concepts from [Jun05]) ........................................................................................................ 11

Figure 4: Detection of sharp features is done by clustering of the normals of the points .... 13

Figure 5: (a) Structured mesh has the same topology as a square grid of triangles; ............ 17

Figure 6: Delaunay triangulation procedure can be thought of as a function that takes a
set of points and outputs a triangulation ............................................................................. 19

Figure 7: The circumcircle of every Delaunay triangle is empty ............................................ 19

Figure 8: Triangular mesh generation by conventional AFT (reproduced from [BK06]) ....... 24

Figure 9: A quadtree .............................................................................................................. 25

Figure 10: Example of Delaunay refinement, where point A is inserted ............................... 27

Figure 11: Edge relaxation: If, for an edge e the vertex is inside the circumcircle of
triangle T then the edge e is swap (concepts from [Lie03]) .................................................. 28

Figure 12: Basic concepts related to a hole in a mesh model ............................................... 31

Figure 13: 1-ring neighborhood information of vertex A ....................................................... 32

Figure 14: Example of the case when the detection of crest/feature points goes
inaccurately at the hole vicinity .......................................................................................... 35
Figure 15: (a) A situation that may need to connect the crest lines together, crest
segment are shown in brown bold; (b) The angles $\alpha$, $\beta$ and $\gamma$ generated by
crest line end-segments and the segment connecting crest line end-points are
used to measure to decide if the connection is necessary (concepts from
[YBS05]). ........................................................................................................... 37

Figure 16: The detected crest lines on the models obtained in our system using
Yoshizawa’s method [YBS05]: Ridge lines are marked in blue, ravine lines
are marked in green, all the crest points (ridge points and ravine points) are
marked in blue. ................................................................................................... 38

Figure 17: The hole boundaries are detected and marked in yellow; The red square
points are the feature points detected on the hole boundaries. ...................... 40

Figure 18: The holes in the model are identified with the yellow boundaries and
partitioned based on the interpolated feature lines that are supposed to pass
over the holes. ..................................................................................................... 43

Figure 19: The Catmull-Rom spline passes through all of its control points. ............ 44

Figure 20: An example of interpolating the feature lines over a hole using Catmull-
Rom spline interpolation: the detected crest line and detected feature point
are colored in red; the crest points adjacent to the detected feature points are
colored in purple; the interpolated feature line is colored in blue, the
interpolated feature points are colored in green............................................... 46

Figure 21: Example of projection planes for holes on a bunny mesh model. ............... 47

Figure 22: Deriving a normal for a hole ($N$) from the boundary triangle normals. ....... 48
Figure 23: Projection of a hole boundary onto the projection plane and the rotation transformation of the projection plane to parallel to $XY$-plane. ........................................ 49

Figure 24: Moving vector $A$ to $Z$-axis. ................................................................. 50

Figure 25: The two mesh models are used for the experiments with our system user-guide feature sensitive hole filling system: (a) The stripped fan disk model; (b) The moai model. .......................................................................................................................... 53

Figure 26: (a) The input stripped fan disk model with a hole on the convex sharp edge; (b) Our hole filling result without the user intervention to correct the detected crest points at the hole area; (c) Our hole filling result with the user intervention to correct the detected crest points at the hole area; ......................................................... 54

Figure 27: (a) The input moai model with two holes and concave sharp edges; (b) Our hole filling result is obtained in our system with user intervention to pair the feature points. ................................................................................................................ 54

Figure 28: (a) The input stripped fandisk model with a convex corner hole; (b) Our final hole filling result with polygonal presentation, hole patch is color in purple; (c) Flat shaded render of the final result model.......................................................... 55

Figure 29: The closer views at the hole area of the result in Figure 44 ....................... 55

Figure 30: (a) The input stripped fandisk model with a concave corner hole; (b) Our final hole filling result with polygonal presentation, hole patch is colored in green; (c) (d) Flat shaded render of the final result model ....................................................... 56

Figure 31: (a) The input moai model with a convex corner hole; (b) Our final hole filling result with polygonal presentation, the hole patch is colored in red; (c) Flat shaded render of the final result. ................................................................. 57
Figure 32: Hole-filling results: (a) The input mesh model with 3 holes; (b) The result obtained by using method in [Lie03]; (c) The result obtained by using method in [ZMML06] ................................................................. 58

Figure 33: Applying the hole filling algorithm in [CCL05] to the holes at the corners. (a) A original mesh model with a convex corner hole; (b) The result after filling the hole in the model in (a); (c) A original mesh model with a concave corner hole; (d) The result after filling the hole in the model in (c) (reproduced from [CCL05]). ................................................................. 58

Figure 34: A moai model with 3 types of holes: (a) The original mesh model with 3 kinds of holes; (b) The final result obtained by our system with the highlighted patches; (b) Flat shaded render of the final mesh result................. 59
### Glossary of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>Two-Dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>Three-Dimensional</td>
</tr>
<tr>
<td>AFT</td>
<td>Advancing Front Technique for mesh generation</td>
</tr>
<tr>
<td>CDT</td>
<td>Constrained Delaunay Triangulation</td>
</tr>
<tr>
<td>Circumcircle</td>
<td>Circumscribing Circle</td>
</tr>
<tr>
<td>Circumsphere</td>
<td>Circumscribing Sphere</td>
</tr>
<tr>
<td>Coin3D</td>
<td>High-level 3D graphics toolkit for developing cross-platform real-time 3D visualization and visual simulation software</td>
</tr>
<tr>
<td>DT</td>
<td>Delaunay Triangulation</td>
</tr>
<tr>
<td>FEM</td>
<td>Finite Element Method</td>
</tr>
<tr>
<td>MFC</td>
<td>Microsoft® Foundation Classes</td>
</tr>
<tr>
<td>OT</td>
<td>Optimal Triangulation</td>
</tr>
<tr>
<td>PSLG</td>
<td>Planar Straight Line Graph</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>RMT</td>
<td>Regular Marching Tetrahedral</td>
</tr>
<tr>
<td>VTK</td>
<td>The Visualization Toolkit</td>
</tr>
</tbody>
</table>
Chapter 1      Introduction

In this chapter we introduce the objectives of this thesis, the problem we address and an overview of the solutions we implement. Then we present the organization of this document.

1.1 Motivation

3D computer models of real life objects can be obtained by several ways such as 3D scanning devices, or computer-aided design software (Autodesk Maya, 3DS Max, etc.). A common scenario, especially when dealing with 3D shapes obtained from 3D scanning, is to have incomplete surfaces. These appear in areas where the object geometry occludes the scanning device, notable examples when scanning human bodies include the area under the chin, armpits or between the fingers, hence limiting the information obtained. Thus many post processing techniques are needed to be applied onto the raw models obtained from the scanning devices before being able to use them as the input of design or animation applications. The repair of incomplete polygon meshes is a fundamental problem in the reconstruction of 3D models in the field of computer graphics.

One of key aspects of reconstruction of 3D models is hole-filling. This is to complete the shape of the 3D object where surface information is missing. This is essential for a wide range of applications such as computer animation, pattern recognition, or character design. Hole-filling techniques aim to keep the filled surface continuously and smoothly fitted at the boundary of the hole to conform to the shape of the original model. Although there is a large body of research on hole filling, very little attention has been devoted to the problem of recovering fine features of the object, like the sharpness of the edge geometry. Most
research focuses on automatic methods that require performing complex optimization processes. While the models obtained are complete and hole-free, interpolation algorithms fail to preserve fine details, ignoring sharp edges.

Due to the complexity of the region where holes are generated, automatic model modification methods may not give satisfactory results in dealing with holes. Complex optimization frameworks are computationally expensive and processing large and complicated models is a time consuming task. Despite of the great computational overhead, fine features in models are not recovered. Since there are potentially several possible results for the surface recovery process, the user should have the ability to influence the quality of the output surface. We believe user intervention efficiently helps to reduce the implementation effort, to give better visually plausible results and to enhance the versatility of the system since the user would have the ability to choose the desired topology of the filled mesh.

We are motivated by the need of a hole filling system that is able to aesthetically recover the fine geometry features of the 3D models, especially the sharp features, with some simple guidance by users at the hole locations using a real time Graphics User Interface (GUI). Our goal is to develop a system that can repair the holes of the 3D models and, at the same time, aesthetically preserve the sharpness of the model at the hole locations with the aid from the user intervention.

1.2 Problem Statement

In this thesis we address the issue of developing a technique for filling holes in 3D triangulated models that is able to recover the sharp features of the original model at the hole locations. The requirements of such a scheme are:
• The holes should be completely filled;

• The fine geometry features of the 3D models at the hole locations should be aesthetically recovered, especially the sharp features, with some simple guidance by users (real time graphics user interaction).

• Since there are potentially multiple desirable solutions the user should be able to influence the results, to guide feature curve reconstruction wherever ambiguity exists or results are unsatisfactory, e.g. the scheme should support user intervention to correct the detected salient features at the hole areas if they are not correctly detected.

1.3 Proposed Solution

In this section we briefly describe the solution for the problems stated in section 1.2. The detailed solutions will be discussed in the ensuing chapters.

Our criteria are that the fine features (sharpness) of the model at the hole locations are ensured to be preserved and then recovered. The hole is partitioned into the more planar sub-holes by several ‘feature’ lines to be more efficiently handled in the next steps. We estimate the feature line information in the missing areas out of the salient features information on remaining mesh as it gives information about how to interpolate the missing feature line segments over the holes. We use the crest line detection methods to detect the ridges and ravines of the original models. The intersections of the crest lines with the hole boundaries are then calculated to decide the end points of interpolated feature segments over the holes. The information of the corresponding crest lines are also used for the feature segment interpolation process. The interpolated feature segments are then used to partition the original complex holes into the sub-holes that are more planar, simpler and easier to handle for later processing.
Since there exist multiple desirable solutions, the user should have the ability to influence the output surface. Furthermore, although there are many ways to set the constraints to automatically resolve ambiguous topology problem, there will always be the cases that require high-level knowledge to disambiguate. We believe this knowledge is best provided through the user intervention. In addition, since the results from the automatic salient feature detection are not always accurate, to enhance the quality of the output results we also allow the user being able to visually correct those incorrect detected ones to its right positions. So that it is ensured to get proper salient information for further processing.

We design data structures to store salient information and polygonal-based information of the input mesh model at the loading phase to reduce the processing time significantly in order to allow the user interact in real time with the system.

1.4 Document Organization and System Overview

Figure 1 shows a high-level view of our system framework. The input mode is loaded into two modules: Crest Line Detection and Hole Detection. Crestline information helps to find feature points near holes, which will be used later for interpolation. Here the user can interfere the feature point location. Using these points, our system performs a feature line interpolation procedure over the holes. This process defines the expected fine features of the hole geometries and also divides large complex holes into planar ones. For each of these simpler holes, patches are generated by projecting the hole on a 2D plane, performing triangulation and then mapping the triangulated surface back to 3D space. Then the patch is stitched into the 3D model, and it is regularized to make the patch consistent with the original mesh, in order to produce the final repaired mesh model.
In Chapter 2 we present a literature review on hole filling approaches, mesh generation and mesh refinement techniques and a discussion on feature recovering hole filling. In Chapter 3 we present the hole filling system. This includes the solution for detecting the salient features of the model; the hole tracing method; the selection of triangulation method that is suitable for the stated hole triangulation problem; the solution for implementing the friendly and efficient user interface to aid the user intervention phase; and finally the data structures design to handle efficiently the 3D data and to support the real time user interaction. In Chapter 4 we validate our methods by showing our results on a series of sample reconstructed 3D models. Finally, in Chapter 5 we present our conclusions and discuss future work.

Figure 1: The framework of our user-guided feature sensitive hole filling system.
Chapter 2 Literature Review

In this chapter, we present a literature review on topics that are closely related to the research work presented in this thesis, including hole filling approaches, techniques in mesh generation and mesh refinement. In Section 2.1 we briefly describe hole filling methods presented in literature. In Section 2.2 we discuss different approaches for shape sensitive and edge preserving hole filling approaches that have been developed. In Section 2.3 we present about meshes and triangulation techniques. Finally, in Section 2.4 we illustrate techniques of mesh refinement.

2.1 Hole filling techniques

3D hole filling is an active research area, as it is a fundamental part of surface modeling area. Hole-filling is an important phase in surface reconstruction. Hole filling methods aim to restore an incomplete model to its original geometry. Many studies on hole-filling problems have been done so far. Hole-filling can be performed as a post-processing operation (for mesh repairs) applied after surface reconstruction, or it can be integrated into a surface reconstruction algorithm [DMGL02].

The main difficulty in this area is related to the topology. Simple holes can be filled using simple topologies, like a disk topology. However, it is quite common to find holes with irregular shape, as for instance in surfaces obtained by 3D scanning, and simple hole filling techniques fail in this case. There is a large body of research on hole filling in meshes, as well as dealing with other defects on the object surface. The existing hole-filling methods can be classified in several ways. [Lie03] classified the hole-filling methods into geometric approaches and non-geometric ones; [ZGL07] classified them into volumetric and
mesh-based methods; [ZMML.06] put them into mesh-based and point-based categories and [PR05] separated them into geometric and volumetric approaches.

The most common way is to classify them into volumetric methods and mesh-based methods. The feature sensitive hole-filling approaches may fall into both categories. However, we separate the feature sensitive hole-filling techniques from the others and discuss them in section 2.2 as they are of special interest in our work.

2.1.1 Volumetric Approaches

Volumetric methods are commonly used for surface reconstruction because of their robustness while dealing with complex geometric features. The main idea of most volumetric approaches is first to obtain the surface implicit as the boundary between volumetric regions by representing the model by a set of discrete volumes, named voxels, then to generate a sign for each voxel to indicate that whether it is inside or outside the surface of the geometry. The signs are generated by using the distance map of each point on the geometry. When all of the voxels are assigned signs, the implicit surface is finally obtained by extracting the contour of the volumetric grid. Volumetric methods are usually guaranteed to produce manifold non-interpenetrating surfaces.

In [CL96], a surface reconstructing technique from range image is presented. The hole-filling phase is integrated in the surface construction algorithm. The system uses a volumetric representation that consists of a cumulative weighted signed distance function. Each range image is scan-converted at a time to a distance function then is combined with the acquired data using an additive scheme. The final manifold is obtained by extracting the iso-surface from the volumetric grid. To fill gaps in the model, they employ space carving

7
and iso-surface extraction to tessellate over the boundaries between regions seen to be empty and regions never observed. This system can produce a model with a plausible surface.

In [DMGL02], a volumetric diffusion method is used to fill the gaps in the 3D models. The algorithm starts by constructing a signed distance function whose zero set defines the observed surface. Initially, this function is only defined near the holes in the surface. Then a diffusion process is applied to extend the incomplete surface description until it forms a watertight model. The diffusion algorithm is identical to the heat equation described in [LL87].

In [Ju04], an inside/outside volume is constructed using an octree grid and the surface is obtained by contouring. The difference between this method and previous methods is the use of space-efficient octree grid instead of a uniform volume grid. This means the input model can be processed at a higher resolution with less space consumption. Figure 2 shows an example of an octree decomposition of the 3D bunny model. The meshes are repaired by contouring the half-edge loops surrounding the holes, filling them based on local constraints. This method is also able to reconstruct the sharp feature using dual contouring technique described in [JLSW02].

![Octree Decomposition](image)

Figure 2: An octree decomposition of the bunny model (reproduced from [PR05])
The disadvantage of the previous approach is that the original surface is only approximated and even the topology of the input surface may be changed. To overcome this problem, Polodak and Rusinkiewicz [PR05] used a global optimization approach. Instead of growing local patches from hole boundaries until they connected. An octree decomposition is employed to partition the space into atomic volumes. Then a min-cut algorithm is used to split the graph into inside and outside sub-graphs and to patch the holes simultaneously. In their convention, a volume is atomic if it cannot be intersected by the polygons of the mesh. Then the output model is the union of the interior atomic volumes.

Although volume based approach for hole filling produce plausible results in many examples, they are usually very time consuming and in some cases may even produce incorrect topologies [ZGL07].

2.1.2 Mesh-based Approaches

In mesh-based approaches, the holes are patched by dealing directly with the mesh representation elements, usually points, triangles or tetrahedra. Triangle meshes are the simplest and most effective form of interpolation between surface samples [AFR*03].

In triangle meshes, holes with regular boundaries over a relatively planar region can be easily patched via planar triangulation, which has been described in detail by a number of textbooks and papers [ZGL07]. However, dealing with complex holes is much more challenging, thus many mesh-based hole-filling techniques have been proposed, e.g. filling-hole by triangulation and refinement [Lie03] [Jun05] [ZGL07]; by using radial basis functions [CBC*01] [CLMM05]; by template-based mesh completion techniques [KS05] [KHYS02]; by context-based mesh completion techniques [SACO04], etc. The selection of
hole filling techniques basically depends on the type of the input model, the complexity of the holes on it and the set of geometric features of the 3D objects that needs to be recovered.

In [BK97] a Repair by Shifting Vertices of Polygon (RSVP) system is implemented. RSVP takes a soup of polygons as the input and produces an adjacency structure of the corrected model. The algorithm starts with computing the connected components of the surface. Then the next step is to match each boundary edge with another using an adjacency score. Two boundary edges with minimum score are merged by moving their end points. Holes and open components remaining after the merging process are identified by computing a Jordan curve on the modified surface. Holes are then triangulated by computing the triangulation of a hole with \( n \) edges in \( O(n^3) \). The method attempts to minimize the surface area of the triangulation and to avoid as much as possible long skinny triangles.

To deal with complex holes in 3D meshes, the method in [Lie03] first detects the hole boundaries, then obtains first patches for the holes by doing the triangulation of 3D polygon defined by those boundaries. The triangulation method extends the minimum-area algorithm proposed in [BDE98]. Mesh refinement is applied to the triangulated patches by adapting the edge relaxation algorithm in [PS96]. The final process is done using an umbrella operator to fair the refined triangulation over the hole to estimate the underlying geometry. Generally, the algorithm can produce good results for arbitrary holes in meshes, especially small holes that have a simple topology, but it fails to reconstruct the salient features at the holes area, if any. The \( O(n^3) \) performance of this triangulation method is also a limit to its wide application.

In [WO03], a hole-filling algorithm on surfaces reconstructed from point clouds is presented. The method starts with automatically finding the hole boundary and its vicinity.
Then it computes the tangent plane for the hole vicinity and determines new point positions on that plane by resampling. The surface fitting and new resampling procedures that are based on the moving least square technique presented in [LS86] close up the hole-filling procedure.

In [Jun05], a hole-filling method based on a piecewise scheme is proposed. The method divides a complex hole into several simpler ones by projecting its 3D boundary into its tangent plane. The hole is then split at the intersection points of the projected boundary edges on that plane. All obtained sub-holes are sequentially filled with a planar triangulation. Subdivision and refinement procedures are then applied to smooth the new triangles. The disadvantages of the method are that too many overlaps or twists may make it crash and iterative refinement is a time-consuming process. The method is also not robust for recovering the sharp feature of the mesh at the hole areas. The illustration of the projection performed in this approach is shown in Figure 3.

![Figure 3: Projection of polygonal holes on a plane. (a) Simple hole; (b) Complex hole (concepts from [Jun05]).](image-url)
In [CCL05], a hole-filling method that can recover the sharp features of the 3D mesh at the hole area is presented. In this method, holes are identified first. Implicit surfaces for patching the holes are then created using a radial basis function. The implicit surfaces are triangulated using regular marching tetrahedral (RMT) technique, and then they are stitched to the original 3D model. A feature enhancement process based on Bayesian classification and the *sharpness dependent filter* presented in [CC05] is then applied if there exists any sharp feature on the hole boundary.

Radial basis functions are also used to compute thin-plate interpolations of a hole in scattered height data [CBC*01], as a way of surface reconstruction, which also fills the holes in the surface.

### 2.2 Feature recovering and edge preserving hole-filling

In general, hole filling techniques allow producing a continuous and smooth surface in the boundary of a hole. This should reproduce the original object shape. However, the surface interpolation techniques produce smoothed models, where fine details and sharp edges are lost [CCL05]. So far there are only few hole-filling approaches attempting to preserve and to recover the sharp features of the 3D model. The algorithms in [OBA*03], [SACO04], [AFR03] and [Jun05] employ different post processing techniques, usually refinement and smoothing, to produce smooth results and to improve the sharp features in the repaired mesh model. Although an automatic system is always desirable, dealing with fine features at the hole areas is a challenging task. Even with very complicated optimization engines to get the filling results automatically as in [Jun05], the results are often not adequate. Most of the systems require user intervention [BK97] [HC06] [ZMML06] to obtain the best guess of fine features at the hole areas and to correct the automatic results.
In the approach proposed in [BK97], the system allows users to inspect the automatic results of the first iteration and also to mark the areas to be corrected. The second iteration gets the final results. The approach can produce "intuitively-correct" filling of the holes with the aid of the user.

To preserve the shape of sharp edges and corners, the approach in [OBA*03] employs a multilevel piecewise surface fitting method to represent a mesh model that has fine structures. Local approximation for fitting edges and corners are based on the piecewise quadric surface fitting method. It consists of a number of tests (edge tests and corner tests) in order to determine the type of approximation surface or shape function that should be used. Edges and corners are automatically recognized using the procedure proposed in [KBSS01]. The idea is based on clustering of normals, as demonstrated in Figure 4.

![Diagram of normal vectors](image)

**Figure 4:** Detection of sharp features is done by clustering of the normals of the points.

Sharf et al. [SAC04] proposed a context-based completion method to recover the missing fine details in a repaired hole to conform to the original model. They employ the idea of texture synthesis, by replicating portions of regions from adequate examples. Based on this idea, in their algorithm, the fine structure of the 3D model is recovered by finding a
piece in the original model or in the template models that is similar in shape to replace the initial repaired hole. Thus, this method is particularly efficient for repairing holes in textured mesh model.

Attene at al. [AFRS03] proposed a method to recover the sharp features of 3D mesh model that are lost by reverse engineering or by remeshing processes that use a non-adaptive sampling of the original surface. The algorithm starts by identifying the smooth edges in the mesh model then applying the filters to get the chamfer edges. Each chamfer edge and its incident triangles are subdivided by inserting new vertices. These vertices are calculated so that they lie on intersections of planes that locally approximate the smooth surfaces that meet at the sharp features.

In the method in [CCL05], holes are filled and sharpness is recovered by applying a sharpness-dependent filter. The filter operates based on the distribution of the sharpness values of triangle faces that are in the vicinity of a hole boundary. In this case, the vicinity of the hole boundary is defined as its 2-ring neighborhood. For any triangle face, its sharpness value is computed as the variance of the angles between its normal and each of the normals of the neighboring faces. [CCL05]

The method in [HC06] employs both automatic and interactive methods for hole-filling. A novel hole-filling system is proposed, that makes use of a haptic device. After the hole identification phase, the hole boundaries are smoothed in the interpolation step. This step is to correct boundary topologies and to adjust the boundary edge lengths in order to avoid the uneven distribution of points at the hole boundary. Then the user can decompose those complex holes into simpler ones in stitching process. The sub-holes are then automatically triangulated using regular triangulation methods. The user can repeat the
intervention process until obtaining the satisfactory results. The authors proposed an interesting idea about using haptic for 3D user intervention but the lack of an automatic method to detect the fine feature of the mesh to serve as the guidance for the user is one of the limits of this method.

In the approach in [ZMML06], holes are detected, then triangulated, and sharpness is recovered by crest line fairing. The detected holes are triangulated using the modified minimum-weight triangulation technique presented in [Lie03]. The system makes use of the crest line detection technique in [YBS05] to detect the feature lines in the original mesh. Detected crest lines are then used in region growing and fairing processes to recover the sharp feature at the hole areas. The users are also able to connect some crest lines before the region growing step.

In [CC08], a sharpness-based method is presented for filling holes. The whole algorithm performs in two steps: an interpolation step for filling the hole which produces the first approximation of the final model, and a post-processing step which modifies the approximation model to match the original. The patch for the hole is interpolated using the radial basis function to create a smooth implicit surface to fill the holes. The implicit surface is triangulated using a regularized matching tetrahedral algorithm. Then the triangulated surface patch is stitched to the hole boundary to obtain the repaired model. In the post-processing step, a sharpness-dependent filter is applied to the repaired model to recover its sharp features. In this paper, the sharpness-dependent filter is an improvement of the one presented in [CCL05]. Although the algorithm works quite effectively in repairing the models, the system is difficult to implement.
Our system makes use of the crest line detection method proposed in [YBS05] to detect the ridges and ravines on the whole original geometry. Our method is different from the algorithms mentioned above, as the detected crest lines passing over the hole areas are used to guide the user for pairing the feature points on the hole boundaries. We believe that the interpolation of salient features at the hole area using the crest line information giving the much more accurate results comparing to the use of crest lines for region growing step for salient feature recovery of the method in [ZMML06]. Furthermore, the appearance of feature lines over the holes helps dividing complex holes into simpler ones. This facilitates implementing the hole patching up process. Comparing to the method in [HC06], the user interaction phase of our method is guided much more clearly and intuitively with the help of visible detected crest lines passing over the holes. Most of the hole-filling methods having been proposed do not pay enough attention to the problem of holes at the corners. We show that with limited user intervention, adequate and efficient solutions can be obtained to this challenging problem.

2.3 Mesh generation

We do not intend to cover all previous work about mesh generation approaches since comprehensive surveys are already available [Owe98], [FG00], [TWL00], [Epp01]. Nonetheless, in this section we briefly discuss about the main triangular and tetrahedral mesh generation techniques as triangle mesh generation techniques are heavily used in our approach in the process of patching the holes of the 3D models.
2.3.1 Structured meshes and unstructured meshes

Meshes can usually be categorized as structured or unstructured. A structured mesh is the simplest representation, where each internal node has an equal number of adjacent elements [Owe98]. According to [TWL00], there are two types of structured meshes: geometrically and topologically structured meshes. A mesh is geometrically structured if all of its elements are geometrically alike. A topologically structured mesh is a mesh whose topological structure is isomorphic to that of a geometrically structured mesh. The mesh generated by a structured grid generator typically consists on quads or hexahedra.

In an unstructured mesh, an internal node may have any number of neighboring elements. It has varying local topology and spacing. It is necessary to solve problems with complex geometry boundaries. These meshes are usually represented by triangles and tetrahedra. However, they can also be represented by quads and hexahedra. The polygonal representation of structured mesh and unstructured mesh are illustrated in Figure 5.

Figure 5: (a) Structured mesh has the same topology as a square grid of triangles; (b) Unstructured mesh has varying local topology and spacing.

The main advantage of structured meshes over unstructured are their simplicity. They are less intensive in memory usage, as it is not necessary to store each node explicitly,
since they can be computed. This cannot be done on an unstructured mesh. Also unstructured meshes allow more control over size and shape, and can be easily used for simple finite difference methods. However, a major disadvantage of structured meshes is the difficulty to fit a complex shape [BE92]. For those significant reasons, unstructured meshes have been preferred in various complicated application over structured ones. In our application we consider the cases of both structured and unstructured triangular meshes.

2.3.2 Delaunay Mesh Generation

Delaunay triangulations have become popular for triangulating arbitrary planar domains that are subsequently mapped onto the surfaces due to their properties relevant to the Finite Element Analysis (FEA) [Fie92]. In this section, we mostly discuss Delaunay mesh generation in 2D since the mesh generation technique to create the patches used in our system is based on 2D methods.

2.3.2.1 Delaunay triangulation

In this part, we briefly introduce the traditional Delaunay triangulation and its properties. Since our research is mainly concerned with 3D models, we limit our discussion to the details that are relevant for mesh generation.

The Delaunay triangulation can be defined as follow:

“In two dimensions, a triangulation of a set \( V \) of vertices is a set \( T \) of triangles whose vertices collectively form \( V \), whose interiors do not intersect each other, and whose union completely fills the convex hull of \( V \).” [She97, p.12]
Figure 6: Delaunay triangulation procedure can be thought of as a function that takes a set of points and outputs a triangulation.

"Delaunay triangulation (DT) is a triangulation whose edges and triangles are all Delaunay. An edge or triangle is Delaunay if it has an empty circumsphere, the one that encloses no vertex."[She05, p.11] Figure 6 demonstrates an example of the Delaunay Triangulation obtained from a set of input points in 2D.

In 3D, every edge, triangular face and tetrahedron of the Delaunay triangulation has an empty circumsphere. An example of how this property holds in 2D is shown in Figure 7.

Figure 7: The circumsphere of every Delaunay triangle is empty.
Other important properties of the Delaunay triangulation under different conditions on the set of points described in [BE92]:

If no four points in $P$ are cocircular, then the Delaunay triangulation is unique. Also, the Delaunay triangulation corresponds to the dual of the Voronoi diagram of the set of points. The Voronoi diagram of a set of points $P$ corresponds to a graph where each edge is equidistant to the two nearest vertices in $P$.

Let $S$ be the space enclosed by the convex hull of the set $P$. If there are no four cocircular points in $P$, then the partition of $S$ determined by the dual of the Voronoi diagram contains only convex faces. Moreover, any triangulation obtained by adding edges to triangulate faces with more than three edges becomes a Delaunay triangulation.

The Delaunay triangulation solves the problem of connecting a set of existing points in space. However, generating a mesh is a more complex task, as in general it is required to add points in space, discarding triangles with angles that are too small or too large, or refining poorly shaped mesh elements. Lawson [Law77] proved that the 2D Delaunay triangulation maximizes the minimum angle. However, the minimum angle may still be too small. In other words, the poor quality elements may still appear. Furthermore, the Delaunay triangulation might not conform to the domain boundaries. This means the input boundaries may fail to appear after triangulating. Thus, finding proper methods to treat those problems in the triangulation domain is necessary. According to Shewchuck [She05], both mentioned problems can be solved by inserting additional vertices into the triangulation.
2.3.2.2 Point Insertion

There are several ways to insert points into a triangulation for the purpose of meshing. The first, and also the simplest one, is to define a regular grid, and then to insert points to the domain according to the grid criterion. The second way is to insert the points to the circumcircle or circumsphere centers [Che89] [Rup92]. In 3D, for tetrahedral mesh generation, the points can be inserted at the centroids of the tetrahedra [WH94].

The method for inserting points, proposed in [Reb93] that oriented to satisfy local size criteria, employs the concept of Voronoi segment. The latter is a line segment between the centers of two adjacent triangles circumcircles. In 3D this would correspond to a line segment between the centers of the circumspheres of adjacent tetrahedra. Points are inserted in the Voronoi segment. Meshes generated by this way in general are very regular, where vertices are adjacent to six triangles.

Another method, proposed in [MW95] uses the concept of advancing front. An advancing front is defined at the object boundary. In an incremental process, facets are examined to find the best location for the new point. Then the point is inserted, connected locally and the front advances towards the interior of the object. As in the case of insertion on the Voronoi segments, triangular meshes computed with an advancing front are very regular. These meshes are also well aligned with the object boundary. Another different approach is proposed in [BHSG95]. In this method, points are inserted along edges in a recursive process that finishes when a background sizing function is satisfied. These points are generated by marching in the existing edges, and inserted incrementally. Nodes that are too close to neighbors are discarded, and the process continues. Most other methods developed are very similar to the ones above.
2.3.2.3 Boundary Constrained Triangulation

In many applications, the usual input for the 2D mesh generation is not merely a set of vertices, but it often includes some given edges or triangulation. There may be a requirement to maintain a certain existing surface triangulation. In most Delaunay approaches, before internal nodes are generated, a simple Delaunay triangulation for the set of initial nodes is produced. The initial edges are recovered later by ad-hoc techniques.

Edge recovery in 2D can be performed in a more or less straightforward way. By iteratively swapping triangle edges, the triangulation can be recovered. Vertices are inserted when a swap cannot resolve the edge.

In 3D an additional facet recovery step is required. Surface facets have to be maintained, therefore, after the edge recovery step, additional transformations are performed. These consists mainly by swapping three adjacent tetrahedral at an edge for two. In this process, additional vertices may have to be inserted in order to successfully perform facet recovery. In another approach proposed in [WH94], additional vertices are temporarily inserted directly into the triangulation wherever the surface edge or facet cuts non-conforming tetrahedral. Once the surface facets have been recovered, those additional vertices are then deleted.

2.3.3 Advancing front methods

Advanced front methods are another mesh generation techniques focusing on complicated boundary geometries. In the advancing front methods, the boundary of the mesh is divided into edges, or triangles, depending on the dimensions of the problem. This boundary represents the initial front. Then edges, or triangles, are generated, from the
boundary inwards. Then, the process is repeated, by updating the boundary according to the recently inserted elements. [She97]

The detailed steps of the conventional Advancing Front Technique (AFT) for mesh generation were described in Banu and Kumar’s paper [BK06] as follow:

i. The domain to be meshed is modeled using boundary curves defined by NURBS;

ii. The boundary is discretized into linear segments to form the initial front $Γ$ as shown in Figure 8(a);

iii. An active edge $AB$ is selected from the front $Γ$. Generate an element by inserting a new node $N$ or by taking one of the nodes in the front $Γ$ as shown in Figure 8(b);

iv. Update the front $Γ$ and it advances further into the domain, illustrated in Figure 8(c);

v. The node and element generation is successive and the steps 3 and 4 are repeated until no edge is left available for the element generation as shown in Figure 8(d).
An element is formed by inserting a new node $N$ provided the following conditions are satisfied, referring to Figure 8(b):

i. Edges formed say $AN$ and $BN$ should not intersect with any of the edges in the advancing front $\Gamma$;

ii. Triangle $ABN$ does not contain any existing node in the mesh;

iii. $N$ lies interior to $\Gamma$.

Advancing front methods can be applied to different scenarios, in a variety of applications different than from the above.
2.3.4 Quadtree and Octree methods

Several methods have been developed using quadtrees and octrees. A quadtree is a data structure that recursively partitions a region into squares, which are aligned with the axis. The top level square is called root and encloses the whole input Planar Straight Line Graph (PSLG). Each square in the structure is further divided into four children squares, recursively. Octrees are the generalization of quadtrees to three dimensions; each cube in an octree can be subdivided into eight cubes. An example of quadtree divisions is shown in Figure 9.

![Figure 9: A quadtree.](image)

A leaf square is a square that has no child. Other square are called internal-squares. The number of the leaf squares of quadtree $T$ is regarded as the size of the quadtree, denoted by $size(T)$. The depth of a square $s$ in $T$, denoted by $depth(s)$, is the number of splittings needed to generate $s$ from the top square. The depth of top square is 0. Two leaf squares are neighbors if they have a segment in common. An important concept is that of a balanced quadtree. A quadtree is balanced if and only if for any two neighbor leaf squares $s_1$ and $s_2$, $|depth(s_1) - depth(s_2)| \leq 1$. 

25
2.4 Mesh refinement techniques

Meshes generated by mesh generation algorithms usually have not the required quality, and improvements are needed. For this purpose, mesh post-processing techniques come into the scene. Basically the post-processing operations for meshes include smoothing and refinement. We will mainly discuss mesh refinement techniques in this section.

We refer to refinement as the operations performed on a mesh to reduce the size of the elements and achieve a greater resolution, or point density [She97].

2.4.1 Triangle/Tetrahedral Refinement

Although there are certainly more methods defined, three of the principal methods for triangle and tetrahedral refinement include: edge bisection, point insertion and edge relaxation.

2.4.1.1 Edge Bisection

According to S. Owen [Owe98], edge bisection is the technique that splits the edges in the triangulation. In two dimensions, the two triangles that are adjacent to the edge are split into four triangles. In three dimensions, any tetrahedron sharing the edge to be split is also split. [Riv97] proposes a recursive algorithm for bisecting a triangle in need of refinement by adding a diagonal from the opposite vertex to the midpoint of the longest edge the refinement for its neighbors is in the same way.

One simple way to achieve a desired nodal density for a mesh is to insert a single node at the centroid of an existing element, dividing the triangle in three or tetrahedron in four. This method does not generally provide good quality elements, particularly after several iterations of the scheme. To improve the results, a Delaunay refinement approach
can be used. This consists on deleting the local triangles or tetrahedra and connecting the node to the triangulation maintaining the Delaunay criterion. Any of the Delaunay point insertion methods discussed previously could effectively be used for refinement [Owe98].

Figure 10: Example of Delaunay refinement, where point $A$ is inserted.

Delaunay refinement algorithms operate by maintaining a Delaunay or constrained Delaunay triangulation, which is refined by inserting carefully placed vertices until the mesh meets constraints on triangle quality and size. Figure 10 shows an example of Delaunay refinement by inserting the vertices.

Delaunay refinement algorithms are common since they exploit several favorable characteristics of Delaunay triangulations. One such characteristic is that a Delaunay triangulation maximizes the minimum angle among all possible triangulations of a point set [Law97].

The main idea behind all Delaunay refinement algorithms is to eliminate the possibility of producing skinny triangles by inserting vertex at circumcenter (Bowyer-
Watson algorithm) so that all new edges are at least as long as circumradius of $t$ (because $v$ is at center of empty circumcircle) [She05].

2.4.1.2 Edge relaxation

Edge relaxation is used with mesh refinement in order to maintain a regular point density while the process is performed. Edge relaxation is defined in [Lie03, p.04] as follows.

Presented in [Lie03], the procedure to relax an edge as follow: for the two triangles adjacent to an edge, check if the two non-mutual vertices of these triangles with respect to that edge lie outside of the circumsphere of the opposing triangle (Figure 11). If this test fails, the edge is swapped.

[Lie03] refined the triangulation into a mesh by adapting an algorithm given by [PS96]. The basic idea is to compute edge length data for the vertices on the hole boundary and to diffuse these values into the interior of the patching mesh, subdividing triangles to reduce edge lengths, and relaxing interior edges to maintain a Delaunay-like triangulation.

Figure 11: Edge relaxation: If, for an edge $e$ the vertex is inside the circumcircle of triangle $T$ then the edge $e$ is swap (concepts from [Lie03])

28
2.4.2 Quad/Hex Refinement

As quad and hex meshes are basically structured, the point insertion and edge bisection methods are generally not applicable. The main technique used for quad and hex refinement is to decompose the elements based on a set of predefined templates. The quadtree refinement scheme presented in [TW99] consists on three major steps as follow:

i. Adaptive refinement: iteratively and adaptively perform quadtree refinement by splitting: if a leaf square contains two non-neighbor segments from the input domain, then split it into four equal squares.

ii. Balancing: perform additional splitting until the quadtree is balanced: if a leaf square is more than twice as large as any of its neighbors, split square into four equal size squares.

iii. Local warping: locally match the corners of the leaf squares with the vertex points of the domain and the end-points of the divide-input-segments, and triangulate the point set.
Chapter 3  User-guided Feature Sensitive Hole Filling

In this chapter we present the details of our framework for our user-guided feature sensitive hole-filling system. Our hole filling algorithm can fill up the holes of a model and aesthetically recover the sharpness of the model at the hole areas, if any. This includes the solution for efficient preservation of sharpness properties of the model during the hole filling process; the solution for implementation of a user-friendly interface to aid the user intervention process and the design of data structures for efficiently handling the complex information processed inside the system in real-time.

3.1 Overview of hole filling

3.1.1 Terminologies

In this section we introduce the basic technical terminology to be used in the remainder of this section, and discuss the basic properties of the geometrical objects we use.

A triangular mesh is defined as a set of vertices and a set of triangles that join these vertices. Two triangles are adjacent if they share a common edge. An edge is adjacent to a triangle if it is part of the border of that triangle.

A boundary edge is an edge adjacent to exactly one triangle. A boundary vertex is a vertex that is adjacent to a boundary edge. A hole is a closed cycle of boundary edges. A boundary triangle is such that has at least one boundary vertex. These concepts are illustrated in Figure 12.
The set of triangles with a common vertex $V$ is denoted the \textit{1-ring triangles} of $V$. Analogously, the set of edges with a common vertex $V$ is the \textit{1-ring edges} of $V$. The same way, all the vertices in the 1-ring edges of $V$, except $V$, correspond to the \textit{1-ring vertices} of $V$. The polygonal area that includes the 1-ring vertices, the 1-ring triangles and the 1-ring edges of a vertex is called the \textit{1-ring neighborhood} of that vertex. These concepts are illustrated in Figure 13.

The normal of a vertex $V$ is computed as the average of the normals of the 1-ring triangles of $V$. Similarly, the normal of a triangle, corresponding to the normal of the plane defined by the triangle, is computed by averaging the normal of its vertices.

The 1-ring neighborhood-based structures are used to support the hole-filling algorithm. We record the 1-ring vertices, 1-ring triangles and 1-ring edges of each vertex in the input mesh.
A vertex is called *singular* if it has more than two adjacent boundary edges. Whenever an edge has more than two adjacent triangles, it corresponds to a *non-manifold edge*. An edge that is not in the boundary is called *interior*. Whenever a mesh has neither non-manifold edges nor singular vertices, it is called a *manifold*.

When we perform the hole filling procedure, the original incomplete mesh will be called the *surrounding* mesh, while the actual mesh used to fill the hole will be called a *patch*. We focus on manifold meshes. Moreover, we consider the case in which there is no island in the holes. This means the surrounding mesh is a *connected* mesh.

The surface holes we concern ourselves can be divided in two categories: *simple holes* and *complex holes*. Simple holes are those that can be filled with a planar triangulation, which is the case when all boundary edges can be projected into a plane, without self-intersection. It is not adequate to fill the complex hole with planar triangulation since there are usually self-intersections when projecting the complex hole boundaries into the 2D plane. Thus in our perspective, we attempt to properly subdivide the complex holes into simple ones in order to fill the holes by planar triangulation.
3.1.2 System overview

The whole system includes eight processing steps as illustrated in Figure 1 in Chapter 1. Initially, the system takes an incomplete 3D triangular mesh model as the input. In this step, all the required information about the vertices, polygons (triangles in this case), 1-ring neighborhood information of all vertices, edges and triangles of the mesh are computed and store in our designed data structures for efficient information processing in later steps. In the next step, the crest lines detection procedure is applied to detect the salient features (ridges and ravines) on the original mesh model. A hole detection process is applied independently on the original mesh to identify the holes to be filled. The feature points are defined as the intersection points of the feature lines (crest lines) with the hole boundaries. Once the feature points are calculated, the user is able to pair the feature points, to visually correct the detected crest lines positions by moving the crest points for the more accurate results, to specify the holes with expected corner shapes or to adjust the interpolated points at the corner holes. Feature lines at the hole areas are then interpolated. Based on those feature lines, the holes are partitioned into smaller ones, called sub-holes, for easier patching while guarantee preserve the feature lines during the process. The patch generation step is then executed to generate patches for the sub-holes. Finally, mesh refinement and fairing may be needed to improve the visual quality of the final results.

3.2 Crest lines detection

The crest lines are the salient surface features defined via the first- and the second-order curvature derivatives. Crest line detection has a significant role in our system since it guides the user to pair the feature points on the hole boundaries and to correctly interpolate the feature lines over the holes. In this part, we present the crest line detection approach
proposed by Yoshizawa et al. [YBY*07], their code\(^{(1)}\) for detecting the crest lines of the 3D mesh models is also re-used in our system for feature lines detection phase.

### 3.2.1 Differential geometry background

Consider a smooth oriented surface \(S\) and denote \(k_{\text{max}}\) and \(k_{\text{min}}\) its maximal and minimal principal curvatures. Let \(t_{\text{max}}\) and \(t_{\text{min}}\) be the corresponding principal directions. Denote by \(e_{\text{max}}\) and \(e_{\text{min}}\) the derivatives of the principal curvatures along their corresponding curvatures directions:

\[
e_{\text{max}} = \delta k_{\text{max}} / \delta t_{\text{max}},
\]

\[
e_{\text{min}} = \delta k_{\text{min}} / \delta t_{\text{min}}.
\]

Following [Thi96] let us call \(e_{\text{max}}\) and \(e_{\text{min}}\) the extremality coefficients. The extremality coefficients are not defined at the umbilical points (\(k_{\text{max}} = k_{\text{min}}\)) since the principal directions are undefined there. The ridges are formed by the closure of points on \(S\) where one of the extremality coefficients vanishes. According to this definition, the umbilical points are ridge points.

The crest lines consist of perceptually salient ridge points. They are distinguished into convex and concave crest lines. The *convex crest lines* are given by

\[
e_{\text{max}} = 0, \quad \delta e_{\text{max}} / \delta t_{\text{max}} < 0, \quad k_{\text{max}} > |k_{\text{min}}|,
\]

while the *concave crest lines* are characterized by

\[
e_{\text{min}} = 0, \quad \delta e_{\text{min}} / \delta t_{\text{min}} > 0, \quad k_{\text{min}} < -|k_{\text{max}}|.
\]

\(^{(1)}\)The code is obtained from http://www.riken.jp/brict/Yoshizawa/Research/Crest.html
The convex and concave crest lines are dual with respect to the surface orientation: changing the orientation turns the convex crest lines into concave one and vice versa.

The proposed crest lines detection method is designed for surfaces with dense triangles meshes. The crest lines are detected by estimating the curvature tensor and curvature derivatives through polynomial fitting, in the same way as in the work by Yoshizawa et al. [YBS05].

It also turns out that in our cases, the mesh objects are usually with holes, the crest lines that suppose to pass over the holes areas are missing after the crest line detection phase and need to be recovered by some way. Furthermore, since there is no surface information at the hole areas the detected crest lines in the hole vicinity are usually go incorrectly comparing to the case when the mesh model is complete. In our algorithm, the detected crest line information is used to interpolate the missing parts. Hence, in order to have the accurate interpolation results it is necessary to correct the crest information at the hole vicinities first before the interpolation is proceeded. User intervention to correct the crest line information is chosen in our method.

![Figure 14: Example of the case when the detection of crest/feature points goes inaccurately at the hole vicinity.](image)
3.2.2 Tracing and thresholding crest lines

This section presents the algorithm by Yoshizawa et al. [YBS05] for detecting the crest points and tracing the crest lines in the 3D mesh model. In order to reduce spurious lines, the thresholding method for filtering the detected crest lines will also be presented.

3.2.2.1 Detection of crest points

As described in [OBS04], once the curvature tensor and curvature derivative $e_{max}(v)$ at each mesh vertex $v$ are estimated, the mesh edge $[v_1, v_2]$ is checked to see if it contains a ridge vertex. $t_{max}(v_2)$ is flipped if the angle between $t_{max}(v_1)$ and $t_{max}(v_2)$ is obtuse: $t_{max}(v_2) \leftarrow -t_{max}(v_2)$, $e_{max}(v_2) \leftarrow -e_{max}(v_2)$. Next the following conditions are checked:

$$k_{max}(v) > |k_{min}(v)| \text{ for } v = v_1, v_2 \text{ and } e_{max}(v_1) \cdot e_{max}(v_2) < 0,$$

where the latter verifies whether curvature derivative $e_{max}$ has a zero-crossing on $[v_1, v_2]$. Finally a simple derivative test is applied:

$$e_{max}(v_i) \cdot [(v_{3-i}-v_i) \cdot t_{max}(v_i)] > 0 \text{ with } i = 1 \text{ or } 2$$

(5)

to determine whether $e_{max}$ attains a maximum on $[v_1, v_2]$.

3.2.2.2 Connecting crest points

Presented in [OBS04], the core idea of the procedure to connect the detected crest points in a mesh model is as follow:

i. If two ridge vertices are detected on edges of a mesh triangle, then connect them together a straight segment.

ii. If all three edges of a mesh triangle contain ridge vertices, then connect these three vertices with the centroid of the triangle formed by the vertices.
It turns out that the procedure may generate several close disconnected crest lines in situations similar to those shown in the left image of Figure 15. The method proposed in [YBS05] improves the connecting procedure with one addition: In order to reduce the fragmentation of the crest lines they inspect the mesh vertices and their 1-ring neighborhoods. For each 1-ring vertex neighborhood containing crest line end-points, if $\alpha \leq \pi/3$, $\beta \leq \pi/3$, $\gamma \leq \pi/3$ then connect two end-points, where $\alpha$, $\beta$ and $\gamma$ are the angles between the end-segments and the segment connecting the end-points, as seen in the right image of Figure 15.

Figure 15: (a) A situation that may need to connect the crest lines together, crest segment are shown in brown bold; (b) The angles $\alpha$, $\beta$ and $\gamma$ generated by crest line end-segments and the segment connecting crest line end-points are used to measure to decide if the connection is necessary (concepts from [YBS05]).

3.2.2.3 Thresholding

Once the full set of crest lines is extracted, a filtering procedure is needed in order to remove spurious lines and select the most perceptually-salient crest line structures.

The strength of a detected crest line is measured by the Mobius-invariant quantity:

$$\int \sqrt{|e_{\max}| + |e_{\min}|} ds$$  \hspace{1cm} (6)

where the integrals are taken over the crest line. The motivation behind using (6) is following: The crest lines are a subset of the ridges which correspond to the edges of
regression of the focal surfaces. The quantity $|e_{\text{max}}| + |e_{\text{min}}|$ computed at a given surface point indicate how far/close a small surface neighborhood around the point is from being a part of a Dupin cyclide. The Dupin cyclides are characterized by the condition that both its focal surfaces degenerate into space curves. Therefore the Dupin cyclides consist of ridge points only and have no creases. Thus (6) can be effectively used to filter out spurious crest lines arising at mesh parts corresponding to planar, spherical, conical, cylindrical, and other Dupin cyclide regions on a smooth surface. Figure 16 shows the examples of crest lines of the two mesh models detected in our system using Yoshizawa’s method [YBS05].

![Figure 16: The detected crest lines on the models obtained in our system using Yoshizawa’s method [YBS05]: Ridge lines are marked in blue, ravine lines are marked in green, all the crest points (ridge points and ravine points) are marked in blue.](image)

3.3 Hole identification

In this stage, our system detects all holes in the given triangular mesh model. In the loading phase, all of the one-ring neighborhood and connected component information of
vertices, edges and triangles of the original mesh model are calculated and stored in several data structures. Hence, in this step, all boundary edges can be easily identified by checking the numbers of their adjacent triangles, i.e. for an edge, if the number of its adjacent triangles is equal to one then that edge is a boundary edge. Its two end vertices are the boundary vertices and its adjacent triangle is the boundary triangle. Once the boundary edge is detected, its two end vertices are used as seeds to trace from the connected boundary vertices. If all the identified points form a closed loop they make up a hole.

3.4 Hole partitioning

Although all simple holes can be adequately filled with planar triangulation, complex holes are not often completely filled with the same triangulation method. Because of its complex geometry and topology, the projection of the boundary of a complex holes on a plane is usually self-intersected, making the incorrectly filled or even unfilled when mapping back from planar triangulation into the 3D space. Therefore, if a polygonal mesh model has complex holes, then an extra step to partition it properly into sub-holes is necessary in order to firmly close them. In this section a new algorithm is proposed to fill complex holes in a piecewise manner. The algorithm divides a complex hole into several simple sub-holes according to detected salient features in the original geometry. Then all the sub-holes are filled with the planar triangulation one by one until the whole complex hole is firmly closed.

Once the crest lines detection and hole detection phases are done. Our hole partitioning phase is processed by the following steps:
3.4.1 Feature points detection

In the convention of our system, feature points are defined as the intersection points between the crest lines, i.e. ridges and ravines, with the hole boundaries. These intersection points are those detected crest points, either ridge or ravine points, that lie on hole edges, this is, boundary edges. Figure 17 illustrates an example of the holes of a mesh detected in our system.

Figure 17: The hole boundaries are detected and marked in yellow; The red square points are the feature points detected on the hole boundaries.

3.4.2 User intervention

After the feature points are detected in the previous step. User intervention is needed

- to pair the feature points to avoid ambiguity for the case there are multiple feature lines passing over the hole;

- to adjust the inaccurately detected crest points to enhance the accuracy and the quality of the final result; and

- to specify the hole at the corner of the object model.
3.4.2.1 The range of movement of a crest point during the user correction

The purpose of adjusting the position of the crest points is to correct the direction of the crest lines passing over the holes. In the paper by Yoshizawa et al. [YBS05], crest points (ridge points and ravine points) lie on the edges of the object models. In practical implementation, since the user is able to adjust the positions of the crest points for better hole filling results. Its range of movement is only within its corresponding edge in the mesh model. In our convention, the corresponding edge of a crest point is the edge in the mesh model on which the crest point initially lies after the crest line detection phase. Thus, in our implementation, we attach the movement of each crest point to its corresponding edge. Every time the user moves the mouse cursor position the projection of its 3D coordinates on the corresponding edge is calculated and displayed in the 3D view that reflects the movement of crest points along its corresponding edge. The problem now reduces to the calculation of the projection of a 3D point onto a line segment. Both the point and the segment are rotated so that the segment is parallel to the ZY-plane in the world coordinate system. This just requires finding a rotation transformation that rotates the segment into a direction parallel to the Z direction. This transformation matrix $T$ is then multiplied with the 3D coordinates of the mouse cursor to find the projection on the edge segment.

The detailed calculation of the transformation matrix $T$ is presented in Section 3.5.1.2.

3.4.2.2 Calculate the initial virtual point at each corner hole

In our convention, corner holes are the holes at the corner areas of the mesh model. They are supposed to have cubic shapes if the surface is complete. As discussed in Section 2.2 in the Literature Review Chapter, most of the hole-filling methods proposed do not offer
or do not offer an efficient solution to the problem of holes at the corners. With a little user intervention, that challenging problem is nicely solved by our method. The procedure to deal with the corner holes is as follows:

For each detected corner hole, two main properties hold:

- The hole must have three feature lines passing over the hole area as if the mesh surface is complete; and
- These three feature lines are supposed to meet at the corner point.

Hence, once its three feature points of the corner hole are calculated, it is necessary to have the fourth point at the corner position to connect those three detected feature points to it. In our system, with the aid of users to specify those three detected feature points, the system then initially generate a corner point so that the user can later adjust it to a visually proper position before the feature lines and feature points at that hole are interpolated. The fourth point is generated by calculating the projection of a detected feature point onto a line segment that is made of one of the two remaining feature points and its adjacent crest point. Figure 18 illustrates the interpolated feature lines at the corner hole.

### 3.4.3 Points interpolation using Catmull-Rom spline

To interpolate the missing feature lines passing over the holes, the following issue should be addressed: since we try to make use of the crest lines information that is automatically detected by the system the interpolated feature lines passing over the hole should be interpolated by the available crest lines and crest points.
Figure 18: The holes in the model are identified with the yellow boundaries and partitioned based on the interpolated feature lines that are supposed to pass over the holes.

A spline is a mathematical representation of a curve. It consists of a series of points, called control points, at certain intervals along the curve, and a function that allows defining additional points within an interval. One requirement for the spline in our case is that the curve passes through all the control points, as they define feature lines and their segments act as the edges in the polygonal mesh model. There are various functions available to approximate a curve and Catmull-Rom spline is the one that satisfy the requirement. In this section we focus on a variety of splines known as the Catmull-Rom spline.

Presented in [Dun 05], a Catmull-Rom spline is defined by a set of points, called control points. One of the features of the Catmull-Rom spline is that the specified curve passes through all of the control points - this is not true of all types of splines.
A new point can be found between two control points. This point is specified by a value \( t \) that represents a proportion of the distance from one control point to the next one, as shown in Figure 19.

![Diagram of Catmull-Rom spline](image.png)

Figure 19: The Catmull-Rom spline passes through all of its control points.

If we assume there is a uniform spacing of control points then, given the control points \( P_0, P_1, P_2 \) and \( P_3 \), we can compute the new point location \( q \), given the parameter \( t \) as follows:

\[
q(t) = 0.5 \begin{pmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}
\]

which is equivalent to:

\[
q(t) = 0.5 \cdot (2P_1 + (-P_0 + P_2) \cdot t + (2P_0 - 5P_1 + 4P_2 - P_3) \cdot t^2 + (-P_0 + 3P_1 - 3P_2 + P_3) \cdot t^3)
\]

This formula gives Catmull-Rom spline the following characteristics:

i. The spline passes through all of the control points as shown in Figure 19;
ii. The spline is $C^1$ continuous, this means that the first derivative of the curve is continuous, meaning that there are no discontinuities in the tangent direction and magnitude;

iii. The spline is not $C^2$ continuous, this is the second derivative of the curve is not continuous. This is because the second derivative is linearly interpolated within each segment, causing the curvature to vary linearly over the length of the segment;

A spline segment is defined using four control points but a spline may have any number of additional control points. Therefore, the curve can be defined as a chain of segments defined by the two segment endpoints, plus two other control points in either side of the endpoints. For example, for a given segment with endpoints $P_k$ and $P_{k+1}$, the segment would be calculated using $[P_{k-1}, P_k, P_{k+1}, P_{k+2}]$.

Notice that to define a segment, points outside the segment are required. Therefore the segments at the extreme points of the spline cannot be calculated. If a spline with control points ranging from $I$ to $n$, the minimum segment that can be formulated is $P_1 \leftrightarrow P_2$, and the maximum segment is $P_{n-3} \leftrightarrow P_{n-2}$. Hence, to define $s$ segments, $s+3$ control points are required.

Figure 20 illustrates our method to interpolate a feature line passing over a hole using the detected crest line information. In our implementation, to interpolate a feature line passing over a hole, the pair of feature points on the hole boundary and their adjacent crest points make four initial control points for the Catmull-Rom interpolation equation (7). Since we attempt to interpolate a feature line that has the point density as consistent as possible to
the original mesh, the value $t$ that appears in equation (7) is approximated in our implementation as follow:

Given a hole $h$ that has $n$ edges on its boundary, denote $\text{length}(e_i)$ the length of a boundary edge $e_i$; denote $a$ the average edge length of the hole boundary. We have

$$a = \frac{\sum_{i=0}^{n-1} \text{length}(e_i)}{n} \quad (9)$$

Denote $d$ the Euclidean distance between the feature points $F_1$ and $F_2$ then we have

$$t = \frac{d}{a} \quad (10)$$

Figure 20: An example of interpolating the feature lines over a hole using Catmull-Rom spline interpolation: the detected crest line and detected feature point are colored in red; the crest points adjacent to the detected feature points are colored in purple; the interpolated feature line is colored in blue, the interpolated feature points are colored in green.

### 3.4.4 Subdivision of a hole by hole boundary tracing

Once all the feature lines at the holes are interpolated, the hole tracing procedure is executed. For each hole, the procedure starts with a vertex on the hole boundary, then it does
the tracing along the hole boundary and its corresponding feature lines to subdivide the original complex holes into smaller, more planar and simpler sub-holes.

3.5 Filling a hole through triangulation

Once all the polygonal holes in the original mesh model have been identified, the boundary edges are then projected onto a projection plane for further triangulation, as shown in Figure 21. In this section, we present the method to find the plane to project the boundary of a sub-hole onto and the triangulation method that is employed in our system.

3.5.1 Projection plane calculation

It can be assumed that there exists a plane $P$ such that the projection of the boundary edges of a polygonal hole on the plane is a bounded domain and that plane should limit the possibility of creating the intersection of the projected boundary on it as much as possible. We present the method to find that plane for each sub-hole as follows.

Figure 21: Example of projection planes for holes on a bunny mesh model.
3.5.1.1 Defining the projecting operator

Presented in [Jun05], the technique to define the projection plane is based on the maximum area vector method. The orientation of the plane is derived from the normalized sum of the normals of the boundary triangles, as shown in Figure 22. Then, we can compute the normal \( N \) of the projection plane \( P \) for a hole as follow:

\[
N = n_{sum} = \sum_{k=1}^{v} n_k
\]  

where \( n_k \) is the normal of the \( k^{th} \) neighboring triangle.

![Diagram of deriving a normal for a hole (N) from the boundary triangle normals.](image)

Figure 22: Deriving a normal for a hole (\( N \)) from the boundary triangle normals.

Figure 23 shows an example of hole boundary and its projections onto the plane \( P \). In order to project the boundary vertices onto the plane \( P \), both the vertices and the plane \( P \) are rotated so that the plane is parallel to the \( XY \)-plane in the world coordinate system. This requires finding a rotation transformation that rotates the normal to the plane \( P \) into a
direction parallel to the Z-axis. The transformation matrix $T$ is then applied to all the vertices on the hole boundary [Jun05].

Let $A$ and $A'$ denote the initial and the transformed positions of a vertex, respectively. We have the following transformation equation:

$$A' = T \times A$$  \hfill (12)

![Diagram](image)

Figure 23: Projection of a hole boundary onto the projection plane and the rotation transformation of the projection plane to parallel to $XY$-plane.

### 3.5.1.2 Calculating the rotation matrix

The calculation to find the rotation matrix for the aforementioned problem is as follow:

Given a vector $A = (u, v, w)$ in the world coordinate system. In order to rotate $A$ parallel to $Z$-axis, we do the two following rotations, as illustrated in Figure 24:
i. Rotate $A$ about $Z$-axis to put $A$ in a plane that parallel to $XZ$-plane.

ii. Rotate $A$ about $Y$-axis to be parallel with $Z$-axis.

Note that it requires that the rotation vector not be parallel to $Z$-axis, else $u = v = 0$ and the denominators vanish. The matrix to rotate a vector about $Z$-axis to $XZ$-plane is

$$
R_{xz} = \begin{bmatrix}
\frac{u}{\sqrt{u^2 + v^2}} & \frac{v}{\sqrt{u^2 + v^2}} & 0 & 0 \\
-\frac{v}{\sqrt{u^2 + v^2}} & \frac{u}{\sqrt{u^2 + v^2}} & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$  \hspace{1cm} (13)

The matrix to rotate the vector in the $XZ$-plane to $Z$-axis is

$$
R_{xz2z} = \begin{bmatrix}
\frac{w}{\sqrt{u^2 + v^2 + w^2}} & 0 & -\sqrt{u^2 + v^2}/\sqrt{u^2 + v^2 + w^2} & 0 \\
0 & 1 & 0 & 0 \\
-\sqrt{u^2 + v^2}/\sqrt{u^2 + v^2 + w^2} & 0 & \frac{u}{\sqrt{u^2 + v^2 + w^2}} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$  \hspace{1cm} (14)

Hence, the final rotation matrix to rotate a vector to be aligned with $Z$-axis is

$$
T = R_{xz} * R_{xz2z}
$$  \hspace{1cm} (15)

![Figure 24: Moving vector $A$ to Z-axis.](image)

50
3.5.2 Triangulating the projected hole boundaries

In our implementation, for each hole, once its boundary in the original mesh model are projected onto its corresponding tangent plane, the projected vertices are used as the input for the constrained Delaunay triangulation of the VTK toolkit to get the patch mesh for the hole in 2D. The procedure of mapping back to 3D space of the patch mesh is done by applying the topological structure of the constructed 2D triangulation to the original 3D boundaries.

3.6 The summary of hole-filling algorithm

In the previous sections we discussed the components of our framework and elaborated on the different aspects. Now we are able to outline the overall algorithm as follow:

Algorithm (User-Guided Feature Sensitive Hole-Filling)

i. Detect crest lines and identify the holes of the original mesh model;

ii. Calculate the feature points on the hole boundaries;

iii. User intervention: to correct the crest lines, to pair the feature points, to specify the corner holes;

iv. Interpolate the feature lines passing over the holes using the Catmull-Rom spline interpolation;

v. Partition the holes into several sub-holes according to the interpolated feature segment;

vi. Apply planar triangulation algorithm to each sub-hole;

vii. Apply subdivision algorithm to the newly created triangulation;

viii. Update topology information.
Chapter 4 Results and Validation

In this chapter we present the results of our system with several test cases on mesh models with holes as well as validation by comparing with the original hole-free model. We also compare our results with others. The visualization system Hole3D was developed as the implementation to demonstrate our user-guided feature sensitive hole filling system introduced in this thesis. The visualization and user interface were implemented in MS Visual Studio 2005 Development Environment with Coin3D (a high-level 3D graphics toolkit for developing cross-platform real-time 3D visualization and visual simulation software), VTK (the Visualization Toolkit) and MFC (Microsoft® Foundation Classes). The programming language used is C++.

4.1 Experiment results

We now demonstrate how the proposed algorithm can be used to reconstruct hole regions. Basically, the test cases are processed in the two mesh models, the moai\textsuperscript{[2]} model and the stripped fandisk\textsuperscript{[3]} model as shown in Figure 25. They are the typical models for experiments in hole filling researches. Many possible cases of hole are created and filled using our system to verify its effectiveness.

\textsuperscript{[2]}The model is obtained from http://www.eps.hw.ac.uk/~ab226/soft/ply/gallery.html
\textsuperscript{[3]}The model is obtained from 3D Meshes Research Database by INRIA Gamma http://www-roc.inria.fr/gamma/download/
Figure 25: The two mesh models are used for the experiments with our system user-guide feature sensitive hole filling system: (a) The stripped fan disk model; (b) The moai model.

As it is shown in Figure 26, with the hole at the *convex sharp edge* our system can achieve the proper results. Figure 26(b) shows the hole filling result obtained automatically by our system without a user correcting the detected crest lines. Better result can be obtained with user interaction to correct the detected crest lines before interpolating the feature lines passing over the hole as shown in Figure 26(c).
Figure 26: (a) The input stripped fan disk model with a hole on the convex sharp edge; (b) Our hole filling result without the user intervention to correct the detected crest points at the hole area; (c) Our hole filling result with the user intervention to correct the detected crest points at the hole area;

Figure 27(a) shows the input moai model with two holes and concave sharp edges at the neck area. The sharp features there are recovered properly using our method. The patches stitched to the hole areas are marked with the yellow boundary. Although we do not implement mesh refinement and fairing techniques in our system the patch is adequate to complete the model in the expected way.

Figure 27: (a) The input moai model with two holes and concave sharp edges; (b) Our hole filling result is obtained in our system with user intervention to pair the feature points.
Figure 28 shows the experiment with the fan disk model at a *convex corner* hole (Figure 28(a)). The final mesh model after applying our feature sensitive hole filling algorithm is displayed in Figure 28(b),(c). The sharp features at the corner hole are recovered efficiently.

Figure 28: (a) The input stripped fandisk model with a convex corner hole; (b) Our final hole filling result with polygonal presentation, hole patch is color in purple; (c) Flat shaded render of the final result model.

Figure 29: The closer views at the hole area of the result in Figure 28.

Figure 30 shows the result of filling a fandisk model with a *concave corner* hole (Figure 30(a)). The final mesh model after applying our feature sensitive hole filling algorithm are displayed in Figure 30(b)(c). The sharp features at the corner hole are recovered aesthetically.
Figure 30: (a) The input stripped fandisk model with a concave corner hole; (b) Our final hole filling result with polygonal presentation, hole patch is colored in green; (c) (d) Flat shaded render of the final result model.

Figure 29, Figure 30 and Figure 31 show that our system can produce excellent results for filling the holes at the corners. The mesh quality of the patches could be improved to make them consistent with the original mesh quality by applying mesh refinement techniques.
Figure 31: (a) The input moai model with a convex corner hole; (b) Our final hole filling result with polygonal presentation, the hole patch is colored in red; (c) Flat shaded render of the final result.

4.2 Comparison to other algorithms

In this section, we demonstrate the robustness of our system by comparing our experiment results with the results presented in the papers [ZMML06] and [CCL05]. As shown in Figure 32, there are three holes in fandisk mesh model, which has one, two and three ridges passed through them (Figure 32(a)). As presented in [ZMML06], the method in [Lie03] can only close the holes (Figure 32(b)), and the method in [ZMML06] has better result but the geometry at the corner hole is not recovered properly (Figure 32(c)). Figure 33 demonstrates the hole filling results for convex and concave corner holes obtained after applying the **sharpness dependent filter** hole filling method in [CCL05].
Figure 32: Hole-filling results: (a) The input mesh model with 3 holes; (b) The result obtained by using method in [Lie03]; (c) The result obtained by using method in [ZMML06] (reproduced from [ZMML06]).

Figure 33: Applying the hole filling algorithm in [CCL05] to the holes at the corners. (a) A original mesh model with a convex corner hole; (b) The result after filling the hole in the model in (a); (c) A original mesh model with a concave corner hole; (d) The result after filling the hole in the model in (c) (reproduced from [CCL05]).
To have a visual comparison with the above algorithms, we created the holes at the same locations as in the input mesh model in Figure 32 used in [ZMML06]. Indeed, the modified mesh model shows three typical kinds of holes that have sharp features need to be recovered: one hole with one feature line passing over, one hole with two feature lines passing over and one hole at the corner. The results of our hole filling technique are shown in Figure 34. Figure 34 (b) shows the final hole filling result with the hole highlighted in the red color. Figure 34 (c) shows the hole filling result obtained by our system, the patches are stitched to the input mesh as the hole areas to get the final mesh model.

![Figure 34](image)

Figure 34: A moai model with 3 types of holes: (a) The original mesh model with 3 kinds of holes; (b) The final result obtained by our system with the highlighted patches; (b) Flat shaded render of the final mesh result.

The experiment results in Figure 28 in comparison with the ones in Figure 33 (a) (b), the ones in Figure 30 in comparison with the ones in Figure 33 (c) (d) and the ones in Figure 34 in comparison with the ones Figure 32 show that our algorithm can produce better results than the algorithms in [Lie03], [ZMML06] and [CCL05] for filling hole and preserving the sharp features of the mesh geometry at the hole areas.
Chapter 5  Conclusions

We have presented a novel technique for filling holes in 3D triangulated mesh models which is able to recover efficiently the sharp features of the original geometry, producing plausible results which are consistent with the geometry of the original mesh models. Our system identifies both holes and crest lines and uses this information to segment geometrically complex holes into simple approximately planar holes, called sub-holes. The patch meshes that are used to fill those sub-holes are generated by using planar triangulation algorithm for the point set at the hole boundaries. Then these patch meshes are mapped back to the 3D space and stitched to the original model at the hole areas to achieve the final result. The user is able to interact with our system through correcting the crest lines, adjusting the feature points defined by the crest lines and the hole boundaries, pairing the feature points or specifying the corner hole locations. The adjustment of the location of the crest lines by the user results in modification in the shape of the patch mesh stitched to the original model, as holes are filled using different geometric information. To validate our approach, we have tested our technique on different mesh models, and the results show that our methods effectively reconstruct the sharp features. Most approaches for hole filling in literature do not reconstruct these fine details due to the interpolation schemes used. We overcome this limitation by including additional information on the object shape in areas of high curvature and by limited user intervention.

5.1 Contributions

The following contributions were made as the result of performing the research described in this thesis:
- We develop a feature sensitive hole filling technique that uses information about salient features of the mesh geometry to complete its surface, preserving the sharp features consistently with the rest of the mesh model.

- We develop a graphical user interface with visualization that allows the user to influence the hole filling process and see the results of this interaction in real time.

  Our algorithm differs from other existing algorithms in terms of

  (i) The features are taken as the first subject to reconstruct, which eventually drive the feature-definite surface filling process.

  (ii) The user has the ability to design the features while the rest is taken care by the automatic functions. Our results show missing hole features are recovered with high quality while supporting flexibility.

### 5.2 Discussion

Certainly fully automated methods for hole filling have several advantages over a method that requires user intervention, especially in terms of efficiency. However, from the point of view of the complexity of the hole, a fully automated method may not work correctly for holes with a complex geometry. Our research aims to combine manual and automatic methods to improve current hole-filling methods, making this process more flexible, robust and effective.

In his paper, Jun [Jun05] discusses the self-intersection problem when projecting a complicated hole onto a plane. This means the edges on the hole boundary can overlap in the projection. In our system, since the holes are split at the salient feature curves, the sub-holes obtained are already quite planar. In addition, with the use of the tangent plane of the hole
boundary as its projecting plane our approach avoids efficiently the self-intersection of the hole boundary.

Most of the feature sensitive hole filling methods rely significantly on the normals of the vertices around the hole areas to decide whether or not there exist fine features. This makes those methods sensitive to the mesh quality, e.g. the point density, the shape of the triangle and the point distribution. In our algorithm, since the feature curves are interpolated from the salient information detected in the mesh model, user intervention allows to correct the detected crest line information. This enables the whole algorithm to produce the final result quite independently from the quality of the input mesh model.

The core idea of our algorithm using the salient information to recover the sharp feature is simple but effective. Among the existing techniques that have attempted to reconstruct fine features of the original mesh at the hole areas, our hole filling techniques is different since the fine features are taken as the first subjects to reconstruct, which eventually drive the feature-definite surface filling process. Our results show the effectiveness of our method in filling the hole and preserving aesthetically the sharp edges.

In order to obtain high-quality automated results, we note that our method accuracy depends to a great extent on the accuracy of the crest line detection method. We expect that improvements in the crest line detection will produce a higher quality results from a fully automatic procedure based on our approach.

In our implementation, all of the salient and polygonal-based information of the input mesh model are extracted and stored in our designed data structures in the loading phase then further computation is limited to areas near holes. This makes the algorithm efficient to run on large models.
Our system improves the visual quality of the results with respect to previous approaches and provides real-time user interaction. On the other hand, it strongly relies on crest line detection, and therefore it is very sensitive to changes in this geometrical feature. Our system is able to recover efficiently the sharp features, especially when the feature curves or the profile of the sharp edges are close to the cubic splines. However, if the profile of the sharp edges in the input mesh is more complex than cubic splines, the results may not be necessarily accurate and may even be far from the real geometry.

Further mesh refinement and fairing methods may be used to improve the quality of the generated patch meshes. By doing this, the point density and triangle shape in the patch mesh will be consistent with the input mesh.

5.3 Future Research

The research documented in this thesis lays the groundwork for several advancements in future research.

Since the user interface is for interacting with 3D mesh objects, it should be more intuitive to manipulate points and triangles in 3D to modify and repair models with complex holes. To this end a haptic device could be used instead of a common computer mouse. Integration with haptic device plus a study on how to make use of the haptic force feedback to guide the user to the areas that are more likely to be good candidates for defining the feature lines is one possible part we will do in the future.

The distance of the approximated hole patch and the "true surface" is an important criterion for the quality of our results. However, since we commonly do not have access to the "true surface", sometimes we can only use arbitrary methods and criteria to evaluate the new patch. Holes with complex profiles of sharp edges are difficult to be recovered
properly. Further research may focus on how to patch holes with complex profiles of sharp edges. One possible solution is to combine our methods with template-based strategies to enhance the quality of the approximation of the geometry of the patches.

Our hole filling method works properly with holes that do not contain folds in the direction of the projection onto the plane. This property allows to obtain a proper parameterization of the hole boundary. Our method to subdivide a complex hole at the feature curve positions helps to avoid this kind of problem. However, this is not a general solution and future research may be conducted on finding a more general parameterization of the hole boundary in order to deal with this problem in arbitrary shapes.
References


    in repaired 3D mesh models. IEEE Transactions on Visualization and Computer 


    1999.

[Che89] P.L. Chew: Guaranteed-Quality Triangular Meshes. Technical report 89-983, 
    Department of Computer Science, Cornell University, Ithaca, NY, April 1989.

[CL00] C.K. Chui, M.J. Lai: Filling polygonal holes using C1 cubic triangular spline 

    Range Images. In Proc. of SIGGRAPH, New Orleans, USA, 4-9 August 1996, pp. 303- 

[CP01] G. Cong, B. Parving: Robust and Efficient Surface Reconstruction from Contours. 


Lecturer Notes in Computer Science, Vol. 5259/2008, pp. 742-751, Springer

[Riv97] M.C. Rivara: New Longest-Edge Algorithms for the Refinement and/or
Improvement of Unstructured Triangulations. International Journal for Numerical

[Rup92] J. Ruppert: A New and Simple Algorithm for Quality 2-Dimensional Mesh
Generation. Technical Report UCB/CSD 92/694, University of California at Berkely,
Berkely, California, 1992

[SACO04] A. Sharf, M. Alexa, D. Cohen-Or: Context-based Surface Completion. ACM

[Sch96] R. Schneiders: Refining Quadrilateral and Hexahedral Element Meshes. The 5th
International Conference on Grid Generation in Computational Field Simulations,
Mississippi State University, pp.679-688, April 1996.

of Computer Science, School of Computer Science, Carnegie Mellon University, USA,
May 1997.

Course' slides, the thirteenth and fourteenth International Meshing Roundtables,
September 2005.

Distance Field in Adaptive Resolution. Book of Digitally Archiving Cultural Objects,


