SURFACE RECONSTRUCTION USING GAMMA SHAPES

by

YING SUN

KENNETH R. SLOAN, COMMITTEE CHAIR
MARIETTA E. CAMERON
JOHN K. JOHNSTONE
ALAN P. SPRAGUE
SANKARAPPAN GOPALSAMY

A DISSERTATION

Submitted to the graduate faculty of The University of Alabama at Birmingham, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

BIRMINGHAM, ALABAMA

2006
ABSTRACT OF DISSERTATION

YING SUN

ABSTRACT

Surface reconstruction from a set of sample points is important in many applications, including computer graphics, reverse engineering, computer vision and medical imaging. In this dissertation, several surface reconstruction techniques are investigated and the Alpha Shapes method is studied in detail. A new method, called Gamma Shapes, is introduced and implemented for reconstructing surfaces from a set of unorganized sample points. The Alpha Shapes method requires that the data point set be uniformly sampled in order to reconstruct an acceptable surface. Gamma Shapes method is an extension of the Alpha Shapes method with the advantage that the Gamma Shapes method needs only the $xyz$ coordinates of the data points. Gamma Shapes method automatically selects a local factor, called $\gamma$, to indicate the local density information of the sample points. The value of $\gamma$ is used to modulate the $\alpha$ value, which is used in the Alpha Shapes method for reconstructing surfaces. In this work, four ways of determining the value of $\gamma$ are described and the results are analyzed and compared.
I wish to express sincere appreciation to my advisor Dr. Kenneth Sloan for introducing me to computer graphics, giving the direction to the dissertation, encouraging and guiding me throughout my study.

I would like to thank Dr. Marietta Cameron for her previous work on this research. She has been a great source of good ideas and many discussions with her are invaluable to my research.

I also thank Dr. John Johnstone for his support and suggestions, particularly for the knowledge I learned from his Numerical Computing and Geometric Modeling courses and for the time he has spent with me discussing my research and dissertation writing.

I also thank my reading committee, Dr. Alan Sprague and Dr. Sankarappan Gopalsamy for their suggestions and examination of this work.

I thank all my friends who helped me in UAB.

Special thanks to my girl friend Kaihong Mi for her support, affection and encouragement over these years.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xi</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION TO SURFACE RECONSTRUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Applications</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Rapid Prototyping</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Reverse Engineering</td>
<td>5</td>
</tr>
<tr>
<td>1.2.3 Medical Imaging</td>
<td>6</td>
</tr>
<tr>
<td>1.2.4 Quality Control</td>
<td>6</td>
</tr>
<tr>
<td>1.2.5 Virtual Environment Simulating</td>
<td>7</td>
</tr>
<tr>
<td>1.3 3D Scanning Methods</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Data Categories</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Issues On Surface Reconstruction</td>
<td>13</td>
</tr>
<tr>
<td>1.6 The Contributions</td>
<td>15</td>
</tr>
<tr>
<td>1.7 Outline</td>
<td>16</td>
</tr>
</tbody>
</table>
## 2 PRELIMINARY CONCEPTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Points and Vectors</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Convex Hull</td>
<td>20</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Constructing Convex Hulls</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Voronoi diagram and Delaunay triangulation</td>
<td>23</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Introduction to Voronoi diagrams</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Properties of Voronoi diagram</td>
<td>24</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Constructing the Voronoi Diagram</td>
<td>27</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Introduction to Delaunay Triangulations</td>
<td>28</td>
</tr>
<tr>
<td>2.3.5</td>
<td>Properties of Delaunay Triangulation</td>
<td>29</td>
</tr>
<tr>
<td>2.3.6</td>
<td>Incremental-Flip Algorithms for Delaunay Triangulation</td>
<td>31</td>
</tr>
<tr>
<td>2.4</td>
<td>Medial Axis</td>
<td>32</td>
</tr>
<tr>
<td>2.5</td>
<td>Poles</td>
<td>34</td>
</tr>
<tr>
<td>2.6</td>
<td>Simulation of simplicity</td>
<td>35</td>
</tr>
</tbody>
</table>

## 3 PREVIOUS WORK

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Algorithms Related to Voxel Based Spatial Subdivision</td>
<td>37</td>
</tr>
<tr>
<td>3.2</td>
<td>Algorithms Related to Irregular Spatial Subdivision</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>Algorithm Related to Alpha Shapes</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>Algorithms Related to Voronoi Diagrams</td>
<td>48</td>
</tr>
</tbody>
</table>

## 4 A REVIEW OF ALPHA SHAPES

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>General Position</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>Formal Definition of Alpha Shapes</td>
<td>52</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Delaunay Triangulations</td>
<td>53</td>
</tr>
<tr>
<td>4.3</td>
<td>Determining Alpha Intervals</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Limitations of Alpha Shapes Method</td>
<td>56</td>
</tr>
<tr>
<td>4.5</td>
<td>Weighted Alpha Shapes</td>
<td>59</td>
</tr>
</tbody>
</table>

## 5 GAMMA SHAPES

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>62</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Example of surface reconstruction.</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Contact scanning methods</td>
<td>8</td>
</tr>
<tr>
<td>1.3 Non-contact scanning methods</td>
<td>9</td>
</tr>
<tr>
<td>1.4 Surface reconstructed from a set of points of a hand.</td>
<td>14</td>
</tr>
<tr>
<td>2.1 A vector from $p$ to $q$.</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Vector addition.</td>
<td>18</td>
</tr>
<tr>
<td>2.3 Vector subtraction.</td>
<td>18</td>
</tr>
<tr>
<td>2.4 The angle between two vectors.</td>
<td>19</td>
</tr>
<tr>
<td>2.5 The cross product of two vectors.</td>
<td>19</td>
</tr>
<tr>
<td>2.6 A set of points in two dimensions.</td>
<td>21</td>
</tr>
<tr>
<td>2.7 The convex hull of the points set.</td>
<td>21</td>
</tr>
<tr>
<td>2.8 Voronoi diagram of a set of points in $R^2$.</td>
<td>23</td>
</tr>
<tr>
<td>2.9 Two sites $p_1$, $p_2$ and the bisector $B_{12}$.</td>
<td>25</td>
</tr>
<tr>
<td>2.10 Three sites $p_1$, $p_2$ and $p_3$. The bisectors meet at the circumcenter of the triangle.</td>
<td>26</td>
</tr>
<tr>
<td>2.11 Delaunay triangulation of a point set $S$ in 2D</td>
<td>29</td>
</tr>
<tr>
<td>2.12 Two possible triangulations of four points</td>
<td>30</td>
</tr>
<tr>
<td>2.13 Duality</td>
<td>31</td>
</tr>
<tr>
<td>2.14 The medial axis of a rectangle.</td>
<td>34</td>
</tr>
<tr>
<td>2.15 The poles of a site $s$.</td>
<td>35</td>
</tr>
</tbody>
</table>
3.1 An example of an alpha shape of a set of points $S$. 44
3.2 A family of alpha shapes. Data set is from 3D Meshes Research Database by INRIA GAMMA Group. 45
3.3 Three alpha shapes. 45
4.1 If the cutting ball is too big, the details are hidden from the surface. If the cutting ball is too small, holes are left on the surface. 58
4.2 Limitation of Alpha Shapes method. The data set is generated using Matlab. 59
4.3 A weight is assigned to a point. 59
5.1 The radius of the cutting ball changes according to the point density. The $\alpha$ is modulated by the value $\gamma$. 63
5.2 The poles of a site $s$. 66
5.3 Modulated $\alpha$ balls. 72
5.4 Global poles. 73
5.5 The local feature size of $s$ is not necessarily the same as the smallest radius $r$ of the medial balls touching $s$. 73
5.6 The ellipsoids of body data set. 76
6.1 Surfaces generated using gamma shapes. 78
6.2 Surfaces generated using gamma shapes. 79
6.3 Surfaces generated using gamma shapes. 80
6.4 A rabbit data set of 16760 points. 81
6.5 Surface reconstructed using the Gamma Shapes method. 82
6.6 Surface reconstructed using the Gamma Shapes method. 83
6.7 Gamma Shapes method fails when density is low. 84
6.8 The tori surfaces reconstructed using alpha shapes, Local Pole method, Global Pole method, Ec method and Es method. 86
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Alpha Intervals</td>
<td>57</td>
</tr>
<tr>
<td>6.1 Alpha value and the number of triangles in surfaces</td>
<td>81</td>
</tr>
<tr>
<td>6.2 Evaluation of Gamma Shapes methods</td>
<td>86</td>
</tr>
<tr>
<td>B.1 Running time of experimental data, in seconds</td>
<td>115</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION TO SURFACE RECONSTRUCTION

1.1 Problem Statement

In the Computer Aided Design and Manufacturing (CAD/CAM) industry, the digital object models are designed to manufacture physical models. In the computer graphics industry, existing physical objects are used to get digital representations. There are two reverse problems. The former problem has received much more attention and many modeling tools have been developed to help users to design models in computer. The latter problem is also referred as reverse-engineering or, more specifically, 3D scanning [41]. In recent years, digitization and reconstruction methods have been rapidly developed. Fast and accurate scanning devices are available to acquire various properties of a 3D object, such as its shape, color and reflectance properties.

In order to capture the shape of an object, many thousands or millions of points must be sampled from the surface of the object. The resulting set of sample points, by itself, is one form of representation of the physical object. But this kind of representation is not very useful because a variety of useful operations can not be performed on the digitized object. For example, in robot industry, determining whether two robot arms interfere with each other or not is frequently operated. In this case, a continuous surface representation of the object is required; otherwise, unacceptable results may be generated. In computer graphics, it is often necessary to determine where a beam of light enters and exits the object. A surface representation is also preferred in
this case. In addition, visualization requires the representation of a surface to handle texture mapping, illumination, shading, reflection and so on. All of these require efficient and robust algorithms to be available to generate surface representations from the sample points.

The process of connecting the sample points appropriately to reconstruct the surface is known as surface reconstruction in the computer graphics literature. The problem can be stated as follows:

*Given a set of points \( S \), sampled from a surface \( U \) in three dimensions, to construct a surface \( F \) so that the points of \( S \) lie on or close to \( F \) and the surface \( F \) approximates the original surface \( U \).*

This dissertation presents a new method for reconstructing a surface of arbitrary topological type from a set of unorganized points.

Figure 1.1 gives an example of surface reconstruction. Figure 1.1(a) shows a set of unorganized points sampled from a skull and Figure 1.1(b) shows a surface reconstructed from the set of points. Clearly, the surface reflects more faithfully the real human skull than the discrete data points.

In the process of surface reconstruction, the choice of appropriate mathematical and data structural representation of the surfaces is also important for its applicability. The most common choices are triangular and polygonal mesh representations. Since a triangular mesh allows us to express the topological properties of the surface, it is becoming the most popular model representation for visualization and rendering applications. Most known surface reconstruction algorithms use this kind of representation. In this dissertation, we also choose the representation by triangular meshes for our surface reconstruction method.
Figure 1.1: Example of surface reconstruction.

(a) Sample points from a skull. (Headus, Inc.)

(b) Reconstructed surface.
1.2 Applications

The problem of surface reconstruction has a wide range of applications in diverse fields, including rapid prototyping, reverse engineering, medical imaging, quality control, virtual environment simulation.

1.2.1 Rapid Prototyping

With the fast development of both software and hardware in 3D computer aided design and manufacturing (CAD/CAM), rapid prototyping is becoming an industry standard method to manufacture complex objects. Rapid prototyping (RP) is defined as a class of techniques used to quickly fabricate a physical object using three dimensional CAD data. Rapid prototyping has also been referred to as freeform fabrication (FFF), solid free-form manufacturing (SFF) and layered manufacturing (LM).

Rapid prototyping allows engineers to quickly create complicated three dimensional models of their design rather than just two dimensional pictures. The excellent visual effects of three dimensional models help engineers to communicate ideas with coworkers or customers. Rapid prototyping also decreases development time and cost by giving engineering, manufacturing, marketing, and purchasing people a look at the product early in the design process so that mistakes can be corrected and changes can be made while they are still inexpensive.

But with the development of industry, products become more and more complicated in shape. Sometimes, it is impossible or very difficult to design them in computer directly. In order solve this problem, special designers are hired to create the shapes of the desired objects using clay, wood plaster or form rubber. During the development, clay or wood modeling is easier and quicker for the designers to change the shape of the products than designing in computers. The clay or wood models may be transferred into digital representations by 3D scanning. First, 3D points are sampled from the clay or wood models. Then, surface reconstruction techniques can
be used to reconstruct triangle or polygon meshes of the objects from the sampled 3D points.

1.2.2 Reverse Engineering

In traditional CAD design, models are designed before the manufacture of the physical objects. Conversely, reverse engineering is the process of developing a model of an already-existing object. Usually, a physical object is taken apart (a device, an electrical component, etc.) to see how it works in detail by analyzing the object. After that, models are redesigned in computer in order to duplicate or enhance it [51].

Reverse engineering is widely used in such diverse fields as mechanical designs, electronics, automotive, and even software development.

The following are some reasons for reverse engineering a part of product.

• The original design of the product has been lost or never existed.

• The original supplier is unable or unwilling to provide the products anymore.

• The original CAD model is not sufficient to support modifications or current manufacture methods.

For example, an automobile company may want to analyze the features of a competitor’s vehicle parts when its new model comes to market. The company may buy one and disassemble it, and examine the components to learn how they were built and how they work in order to enhance their own vehicles with similar components.

In practice of reverse engineering a part or a product, 3D points are acquired by using laser range scanners or computed tomography scanners. Then a surface of the product can be reconstructed from the 3D points. After that, the surface may be edited or redesigned and exported to CAD for further refinement and analysis.
1.2.3 Medical Imaging

The problem of surface reconstruction is very useful and important in the area of medical imaging. For instance, in a radiation treatment planning system [59], the reconstructed 3D models of human organs enhance the visual feedback mechanisms and enable a more comprehensive understanding of the spatial relationships to physicians so that the radiation beams can be directed accurately. Actually, according to Lorensen and Cline’s opinions, [50], the medical applications of 3D consist of four steps:

1. **Data acquisition.** Some part of a patient is sampled by the medical imaging hardware. Multiple 2D slices of information are generated.

2. **Image processing.** The original image data can be filtered using image processing or manual techniques and structures within the 3D data can be found.

3. **Surface construction.** This step is to construct a surface model from the 3D data. The model usually consists of 3D volume elements (voxels) or polygons. Users can select desired surfaces by specifying a density value.

4. **Display.** The final step displays the created surface using techniques that include ray casting, depth shading, and color shading.

1.2.4 Quality Control

In industry, the design process of a product is often a long and arduous project. The development of a single part can take weeks or months, from design to production for a certain amount of quality is required to be maintained from raw materials to finished product by periodic inspections. Quality control usually requires feedback information about measured defects to further improvements of the process. Before producing the final product, intermediate products can be digitized by 3d Scanning.
Through surface reconstruction and other operations, this intermediate digital model can be compared to the original model to determine where this intermediate model differs from the original one and what should be improved. Repeat this control work by sending feedback information and fixing problems until the quality of the product is acceptable.

### 1.2.5 Virtual Environment Simulating

A virtual environment is a computer generated 3D spatial environment that simulates the real physical world surrounding us. It is widely used in robotics, military, entertainment, computer games and medicine. During building of the virtual environment, real 3D models are often required to create more realistic models in computer. 3D scanning and reconstruction now can be used to digitize those models.

### 1.3 3D Scanning Methods

Numerous methods for acquiring shape information have emerged in recent years. The methods come in two distinct varieties: contact and non-contact methods.

Contact scanners probe the object through physical touch. They use touch probes that consist of jointed arms or pulley-mounted tethers attached to a narrow pointer [21]. When the pointer touches the surface of the object, a contact event is signaled and the position of the pointer is recorded by computing the angles of the arms or the lengths of the tethers. One example of a contact scanner is a coordinate measuring machine (CMM). It can be very precise and is currently a standard in industrial manufacturing.

The disadvantages of contact scanners are:

- Contact with the surface is required. This may damage or modify fragile objects.
- They are slow.
Contact methods

Destructive Non-destructive

CMM Jointed arms Slicing

Figure 1.2: Contact scanning methods

• Usually, it is clumsy to manipulate and requires significant human operations.

Figure 1.2 shows the taxonomy of contact scanning methods.

Non-contact scanners can be further divided into two main categories: active scanners and passive scanners. There are a variety of technologies that fall under each of these categories.

Figure 1.3 shows the taxonomy of non-contact scanning methods.

Active scanners generally operate by emitting energy waves and recording the transmitted or reflected energy. An example of transmissive method for shape reconstruction is computed tomography (CT), sometimes called CAT scan. Very small, controlled amounts of x-ray radiation are passed through the object and different components of the object absorb radiation at different rates. When special films are exposed to the absorbed x-rays, a collection of images of the inside of the body is captured. The collection of images forms a high resolution volumetric description of the density of the space in and around the object. This volume is ready for surface reconstruction or direct visualization. The advantages of this method over reflective methods are: it is not very sensitive to the reflective properties of the surface, and it can capture the internal cavities of an object that are invisible from the outside. But CT scanners are very expensive and accuracy may degrade when there are large variations in material densities. They also may be harmful to people because of the
The reflective scanners may be divided into two more categories - optical and non-optical scanner. Non-optical methods include sonar and microwave radar. These methods measure distances to objects by measuring the time required for a pulse of sound or microwave energy to bounce back from the object. This kind of scanners is typically inexpensive and is capable of operating over very long distances. Thus, these scanners are suitable for scanning large structures like buildings or geographic features. The disadvantages of these scanners are their inaccuracy and low acquisition speed.

Optical scanners cast structured light onto an object and measure the reflections from the object. These optical methods include imaging radar, interferometry, active depth-from-defocus, active stereo, and triangulation. The advantages of optical
scanners are they are safer and less expensive than industrial CT and they are able to generate dense, highly accurate results. The disadvantage is that they can only capture the visible portions of the surface of the object.

1.4 Data Categories

In order to convert the 3D points into useful computer models, it is essential to develop efficient, automatic and reliable algorithms. Different types of approach are motivated by different types of data. The data points acquired from the scanning devices may be divided into the following three categories:

- Volume data points
- Contour data points
- Unorganized data points

Volume data: The volume data are frequently generated from medical imaging technologies such as CT, MRI and ultrasonography. It is a regular array of points in 3D space. The points are arranged on a regular, rectangular grid. The rows, columns, and planes of the grid are parallel to the $x$-$y$-$z$ coordinate system. The grid divides the space into small cubes, called “voxels”. Each voxel has six faces and each face is perpendicular to one of the coordinate axes. The voxels in a volume data set are identical in shape and the data spacing specifies the length in the $x$-$y$-$z$ directions.

The volume data are regular in both geometry and topology and can be implicitly represented. A natural $i$-$j$-$k$ coordinate system can be defined for the volume data. If the number of points in the data set is $n_x \times n_y \times n_z$, the number of voxels is $(n_x - 1) \times (n_y - 1) \times (n_z - 1)$. So a tuple three indices $i$-$j$-$k$ indicates a particular point or voxel. Similarly, two out of three indices defines a line and a single index defines a plane. The simplicity and compactness of representation are desirable features of
volume data. It is very efficient to traverse and compute with the structure. The major disadvantage with volume data is that the dimensions of the data set must be increased in order to obtain greater data resolution. The volume data set requires an $O(n^3)$ increase in memory requirement. Therefore, to resolve a small feature using volume data may require more disk space or computer memory than is available.

The positions of the data points are all fixed at the grid vertex. Each grid vertex has coordinates $(x, y, z)$ and a scalar value is assigned to each vertex. The scalar values represent some measurable properties or independent variables (e.g., color, opacity, density, volume, material, velocity, strength, time) of the real object. When a threshold is given by the users, a surface can be extracted from the data points. Usually, if the scalar value at the vertex is greater than the given threshold, it means that the grid point is inside the surface of the object (or vice versa); if the scalar value is less than the threshold, it means that the grid point is outside the surface. So given different values of thresholds, a family of iso-surfaces can be extracted from the volume data. Many researchers have studied the problem of reconstruction of surface from the volume data. There are relatively well understood methods such as Marching Cubes method [50] and the geometrically deformed models [53].

Contour data: The contour data consist of a collection of points organized along curves arising from the intersection of a surface and a series of cutting planes. Contour data can be obtained by any method that slices an object and then traces its outline. The contours are usually represented by planar polygons, a piecewise linear approximation to the actual contours. The contour data also can be obtained from existing volume data. For example, the volume data taken from the MRI scanner are in the form of a series of images. A set of contours can be extracted from each slab by comparing the value in each voxel in a slice with a threshold chosen to represent the value of a surface of interest. Sometimes, the process may be done manually if the simple thresholding is not adequate. There are some important differences between
contour data and volume data. The information of volume data are determined by the scalar values at the fixed grid vertices, but the information of contour data lies in the positions of the data points. The positions of the data points are less organized than volume data. And the spatial resolution in one dimension ($Z$ - the spacing between cutting planes) is usually much lower than the spatial resolution in the other two dimensions ($X, Y$ - the spacing of the vertices of the polygon (i.e., the points on the curve) describing the intersection of the surface with the cutting planes. 

The reconstruction from contour data can be divided into three subproblems: the correspondence problem, the tiling problem, and the branching problem [52]. The correspondence problem occurs when an object is represented by more then one contour in any of the slices of the data set or when there are multiple objects present in the data. It involves finding the correct connections between the contours of adjacent slices. The tiling problem involves constructing a triangular mesh between the contours of two adjacent slices. The branching problems exists whenever an object represented by $m$ contours in one slice is represented by $n$ contours in an adjacent slice where $m \neq n$ and $m, n > 0$. The problem of reconstruction of surfaces from contour data has been intensely studied and many methods have been developed such as the method of Boissonnat [12, 13], the method of Meyers, Skinner and Sloan [52] and some other methods [18, 32, 45, 47].

Unorganized data: While there are many details still to be investigated, and methods based on volume data and contour data are still applicable in a wide range of applications, we believe the balance has now tipped towards applications and problems based on 3D unorganized data. Unorganized data are usually generated using laser range scanners (LRS). The data are completely unstructured; usually the data points lie on the surface of the object, but sometimes they are simply scattered measurements distributed throughout the space. The data are gathered as the coordinates of the points on the surface of an object without any knowledge or their topological
connectivities.

Surface reconstruction methods based on this type of data need no further information, such as structure in the data, surface orientation or normals to the surface at each data point. In recent years, a lot of work has been done on this kind of data set [5, 22, 27, 29, 42, 54].

Figure 1.4 shows an example of reconstruction from a set of unorganized of points of a hand. The surface is reconstructed by using SuperCoCone method developed by Dey, et al in Ohio State University.

The nature of the available data often dictates the surface reconstruction methods. Sometimes, it is possible to transform one type of data into another. For example, one popular way to deal with unstructured data is to use interpolation techniques to produce a dense grid of data points (the scalar value typically depends on some measure of distance from the original point cloud) - and then use methods appropriate for volume data to extract a surface. [42].

In this dissertation, we introduce a new method, called Gamma Shapes, for reconstructing a surface from a set of unorganized data points sampled from a surface of an object. The output of our method is triangle mesh on the data points and the mesh is faithful to the topology of the original surface of the object.

1.5 Issues On Surface Reconstruction

Data Sampling: A proper reconstruction of surfaces is possible only if the data points are “properly” sampled. Given a set of uniform data points in 3D, a lot of methods are available to reconstruct a surface from the data, such as Edelsbrunner and Mücke’s Alpha Shapes [27, 29] method and Hoppe et al.’s method [42].

But in real life, the data points acquired from the scanners are rarely uniformly dense everywhere on the unknown surface. Actually, the density often varies with the curvature. Some work has been done to solve this problem. Edelsbrunner introduced
Figure 1.4: Surface reconstructed from a set of points of a hand.
the Weighted Alpha Shapes method [25] by assigning weights to data points according to the densities. Teichmann and Capps provided an Anisotropic Density-Scaled Alpha Shapes method to solve this problem.

Similar to Weighted Alpha Shapes, our Gamma Shapes method extends the Alpha Shapes method. Instead of assigning weights to data points, Gamma Shapes methods locally modulates the $\alpha$-ball, which is used in Alpha Shapes to reconstruct the surface.

Robustness: When implementing geometric algorithms, one of the major obstacles is the principal difficulty of combining numerical or geometric data with topological information. The inconsistent results caused by numerical errors can lead to incorrect results, or even, program crashes. It is common to make some simplifying assumptions in order to make the argument and proof simpler. The input data is often assumed to be “sufficiently well-behaved” such that the algorithm works correctly. It is useful to make such assumptions because talking too much about the implementation details may distract the readers from the main concepts of the algorithm. However, when implementing the algorithms, these details are very important to the programmer and have to be dealt with at some point, or the algorithm will never work correctly.

Until a few years ago, this problem was rarely addressed by the computational geometers, who are mostly theoreticians. Alpha Shapes method uses a symbolic perturbation method called Simulation of Simplicity [28] to achieve a robust and stable implementation. This method simulates an infinitesimal perturbation of the points on the level of geometric predicates and relieves the programmer from the otherwise necessary case analysis.

The Gamma Shapes method extends the Alpha Shapes method. So it also uses the Simulation of Simplicity method to make the implementation robust and stable.

1.6 The Contributions

A brief summary of main contributions of this dissertation is included here.
- Gamma Shapes method takes a set of unorganized points in 3D as input. It does not make any assumptions about connectivities of points. Only the XYZ coordinates of the data points are known. No additional structure information or orientation information of the data points are needed.

- Gamma Shapes method extends Edelsbrunner's Alpha Shapes method with the advantage that the Gamma Shapes method can deal with data points which are not uniformly sampled from objects.

- We provide a way to select “best” reconstructed surface automatically from the family of shapes generated by the Alpha Shapes method and Gamma Shapes method.

1.7 Outline

The rest of this dissertation is organized as follows.

Chapter 2 introduces some preliminary concepts emerged in the dissertation.

Chapter 3 gives an extensive survey of previous work of surface reconstruction algorithms.

Chapter 4 presents a detailed tutorial on Edelsbrunner’s Alpha Shapes method. The definition of alpha shapes and how to use alpha shapes to reconstruct a surface are given in this chapter.

Chapter 5 discusses our new method, Gamma Shapes.

Chapter 7 concludes this dissertation and the future work is discussed.

At last, the appendices describes the implementation details and results of Gamma Shapes method.
CHAPTER 2

PRELIMINARY CONCEPTS

In this chapter, we introduce some of the basic terminology and definitions used throughout this dissertation. We also give some background to the topics in this dissertation.

2.1 Points and Vectors

We denote points, elements of three-dimensional Euclidean (or point) space \( \mathbb{E}^3 \), by lowercase letters such as \( p, q \), etc. Let \( S \) be a set of \( n \) points; we denote the size of the set by \( |S| = n \).

**Definition 2.1.1:** A vector \( \vec{v} \in \mathbb{R}^3 \) is defined by a starting point \( p \) and an ending point \( q \in \mathbb{E}^3 \). A vector is represented using a 3-tuple

\[
\vec{v} = q - p; \quad p, q \in \mathbb{E}^3, \quad v \in \mathbb{R}^3.
\]

![Figure 2.1: A vector from \( p \) to \( q \).](image)

Figure 2.1 show a vector pointing from \( p \) to \( q \).

**Definition 2.1.2:** The dot product of vectors \( \vec{u} = (u_1, u_2, u_3) \) and \( \vec{v} = (v_1, v_2, v_3) \)
Figure 2.2: Vector addition.

Figure 2.3: Vector subtraction.

in $\mathbb{R}^3$ is the number

$$\vec{u} \cdot \vec{v} = u_1v_1 + u_2v_2 + u_3v_3$$

The two vectors are said to be orthogonal to each other, or orthogonal vectors if value of the dot product is zero.

**Definition 2.1.3:** The addition of two vectors, $\vec{u}$ and $\vec{v}$, is a vector $\vec{w}$, which is obtained by placing the starting point of $\vec{v}$ on the ending point of $\vec{u}$, and then drawing a line from the initial point of $\vec{u}$ to the ending point of $\vec{v}$, as illustrated in Figure 2.2.

**Definition 2.1.4:** The subtraction of two vectors, $\vec{u}$ and $\vec{v}$, is a vector $\vec{w}$, which is obtained by placing the starting point of $\vec{v}$ on the starting point of $\vec{u}$, and then drawing a line from the ending point of $\vec{v}$ to the ending point of $\vec{u}$, as illustrated in Figure 2.3.

**Definition 2.1.5:** The Euclidean norm of a vector $\vec{v}$ is the number

$$\|\vec{v}\| = (\vec{v} \cdot \vec{v})^\frac{1}{2}$$

**Definition 2.1.6:** The Euclidean distance between two points $p$ and $q$ of $\mathbb{R}^3$ is
the norm of the difference vector $\|p - q\|$.

d(p, q) = \|p - q\|

Then angle between two vectors can be computed from the dot product and the norms of them.

$$\vec{u} \cdot \vec{v} = \|\vec{u}\|\|\vec{v}\| \cos \theta$$

thus,

$$\theta = \arccos \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|}$$

**Definition 2.1.7:** The cross product of vectors $\vec{u} = (u_1, u_2, u_3)$ and $\vec{v} = (v_1, v_2, v_3)$ in $\mathbb{R}^3$ is a vector
$\vec{w} = (u_2v_3 - v_2u_3, u_3v_1 - u_1v_3, u_1v_2 - u_2v_1)$

2.2 Convex Hull

The topics in topology in this dissertation are limited to the concepts related to surfaces and triangulations. Some definitions below are taken from [54, 55].

**Definition 2.2.1:** Given two distinct points $p$ and $q$, the line defined by these two points is the linear combination $(1 - \alpha)p + \alpha q$, $(\alpha \in R)$.

**Definition 2.2.2:** $\alpha p + (1 - \alpha)q$ $(\alpha \in R$ and $0 \leq \alpha \leq 1)$, defines the line segment between $p$ and $q$. The line segment of $p$ and $q$ is usually denoted by $pq$.

**Definition 2.2.3:** Given $k$ distinct points $T = p_1, p_2, ..., p_k$, the point $x = \alpha_1 p_1 + \alpha_2 p_2 + ... + \alpha_k p_k$, with $\alpha_i \in R$ and $\alpha_1 + \alpha_2 + ... + \alpha_k = 1$, is called an affine combination of $T$.

**Definition 2.2.4:** If none of the $\alpha_i$ is negative, then $x$ is called a convex combination of $T$.

**Definition 2.2.5:** $T$ is said to be affinely independent if no point $p_i$ is an affine combination of $T - p_i$.

**Definition 2.2.6:** A domain $D$ is convex if, for any two points $p$, and $q$ in $D$, the segment $pq$ is entirely contained in $D$.

**Definition 2.2.7:** Given a set of points $S$, the convex hull $\text{conv}(S)$ of $S$ is the smallest convex domain containing $S$.

**Definition 2.2.8:** A halfspace in two dimensions is a halfplane: the set of points on or to one side of a line. This notion generalizes to higher dimensions: a halfspace is the set of points on or to one side of a plane in three dimensions, and so on.

**Definition 2.2.9:** Another definition of convex hull: The convex hull of a set of points $S$ is the intersection of all halfspaces that contain $S$.

The convex hull is one of the most important structures in computational geome-
Figure 2.6: A set of points in two dimensions.

Figure 2.7: The convex hull of the points set.
try. It is used in a wide variety of circumstances. The data considered in our research are sets of points in three-dimensional space. For making it easy to understand, an example of a set of data points in two dimensions and its convex hull are shown in Figure 2.6 and Figure 2.7.

2.2.1 Constructing Convex Hulls

Many algorithms have been developed for constructing the convex hull of a set of points. The gift wrapping algorithm was first suggested by Chand and Kapur [17] as a method for finding convex hulls in arbitrary dimensions. This algorithm runs in $O(n^2)$ time. For each hull edge in 2D or each hull face in 3D, it runs in $O(n)$ time. Jarvis’s march algorithm [44] is a two-dimensional specialization of the gift wrapping approach. Graham’s algorithm perhaps is the first algorithm for finding the hull of points in two dimensions in $O(n \log n)$ time [33]. QuickHull algorithm was suggested independently by several researchers in the late 1970s. It was named the QuickHull algorithm by Preparata and Shamos in [55] because of it is similar to the QuickSort algorithm [48]. QuickHull algorithm can achieve $O(n \log n)$ time complexity in the best case if the set of points is randomly distributed. Although QuickHull is generally quick, it runs in $O(n^2)$ time in the worst case.

All the above convex hull algorithms require all of the data points to be present before the construction begins. This condition, in many geometric applications, cannot be met. Some computation must be done when the points are being acquired.

One well-known incremental algorithm to compute the convex hull of a set of points is to add one point at a time, construct the hull of the first $k$ points at each step, and update the hull when the next point is added. The problem turns out to be how to add a single point to an existing hull. The total time used by this algorithm is $O(n^2)$. 
2.3 Voronoi diagram and Delaunay triangulation

In the field of computational geometry, Voronoi diagram and Delaunay triangulation have been found to be among the most fundamental data structures. They represent important geometric information of an object and they have a strong intrinsic connection with each other that will be discussed later in this section.

More than a century ago, the concept of Voronoi diagram was discussed by Dirichlet in 1850 and first appeared in the paper of Voronoi [62] in 1908. Figure 2.8 shows a Voronoi diagram of a set of points in two dimensions.

![Figure 2.8: Voronoi diagram of a set of points in $\mathbb{R}^2$.](image)

2.3.1 Introduction to Voronoi diagrams

For making it simple, we first introduce the idea of Voronoi diagram in two dimensions.

**Definition 2.3.1:** Let $S = p_1, p_2, ..., p_n$ be a set of points in the $\mathbb{E}^2$ plane. These
points are called the sites. The Voronoi diagram of $S$ partitions the plane into $n$ cells, one for each site in $S$, with the property that a point $q$ lies in the cell corresponding to a site $p_i$ if and only if $d(q, p_i) < d(q, p_j)$ for each $p_j \in S$ with $j \neq i$. That is each cell consists of points of the plane closer to the corresponding site than to any other site.

Voronoi diagrams are in common use in numerous fields in science and real life. For example, Voronoi diagrams are used in ambulance service allocation. When an emergency phone call is received, the Voronoi diagrams are used to decide which hospital should respond to. Voronoi diagrams also are used to solve other geometric problems such as finding nearest neighbor, computing minimum spanning tree and traveling salesman problem.

### 2.3.2 Properties of Voronoi diagram

Before introducing the properties of Voronoi diagram, we make the following two assumptions. If these two assumption are true, we say the set of points is in general position in two dimensions.

**Assumption 1**: No three points of the original set $S$ are on the same line.

**Assumption 2**: No four points of the original set $S$ are on the same circle.

We denote the Voronoi diagram of a set of points $S$ by $V(S)$. The cell that corresponds to a site $p_i$ is called Voronoi cell, denoted by $V(p_i)$. The vertices of the Voronoi diagram are called Voronoi vertices. The line segments of the diagram are called Voronoi edges.

**Definition 2.3.1**: For two points $p_i$ and $p_j$ in the plane, we define the bisector of $p_i$ and $p_j$ as the perpendicular bisector of the line segment $p_ip_j$.

Consider just two points $p_1$ and $p_2$ in the plane. Let $B_{12}$ be the bisector of $p_1$ and $p_2$. Then every point on $B_{12}$ is equidistant from $p_1$ and $p_2$. See Figure 2.9.

The bisector divides the plane into two half-planes. We denote the half-plane that
contains \( p_i \) by \( h(p_i, p_j) \) and the other half-plane that contains \( p_j \) by \( h(p_j, p_i) \).

For three points \( p_1, p_1, \) and \( p_3 \) in the plane, the bisectors \( B_{12}, B_{23} \) and \( B_{31} \) meets at the circumcenter of the triangle \((p_1, p_2, p_3)\). The Voronoi diagram appears like Figure 2.10. The \( V(p_1) \) is the intersection of half-planes \( h(p_1, p_2) \) and \( h(p_1, p_3) \).

\[ V(p_i) = \cap_{i\neq j} h(p_i, p_j). \quad (2.1) \]

**Theorem 2.3.2.1:** Every vertex of the Voronoi diagram is the intersection of exactly three edges of the diagram.

Preparata and Shamos give the proof in [55]. This theorem also tells that each
Voronoi vertices is a center of the circle defined by three points of the original set. For a Voronoi vertex $v$, we denote by $Circle(v)$ the above circle.

**Theorem 2.3.2.2:** The circle $Circle(v)$ contains no other points of $S$.

Proof: Referring to Figure 2.10, let $p_1$, $p_2$ and $p_3$ be the three points of $S$ determining circle $Circle(v)$. Assume that there is another point $p_4$ of $S$ is inside $C(v)$. $p_4$ is closer to $v$ than to any of $p_1$, $p_2$ and $p_3$. So $v$ must lie in $V_4$ and not in any of $V_1$, $V_2$, or $V_3$, by the definition of a Voronoi cell. But this is a contradiction since $v$ is the common point to $V_1$, $V_2$, and $V_3$.

**Theorem 2.3.2.3:** Voronoi cell $V_i$ is unbounded if and only if $p_i$ is a point on the boundary of the convex hull of the set $S$.

See Preparata and Shamos’ proof in [55].

This theorem also implies another way to construct the convex hull from the Voronoi diagram by connecting all these unbounded sites. Actually, if the Voronoi
diagram is available, the convex hull can be easily constructed in linear time [55].

2.3.3 Constructing the Voronoi Diagram

Because of the wide application of the Voronoi diagram, researchers have been motivated to invent a variety of algorithms to construct it.

In this section, we will superficially discuss four algorithms:

**Intersection of Half-planes:** A naive approach to the construction of a Voronoi diagram is to construct its cells one at a time. We know that each Voronoi cell is the intersection of \( N - 1 \) half-planes. The intersection of \( n \) half-planes may be constructed in \( O(n \log n) \) time with a divide-and-conquer algorithm, thereby doing this for each site would cost \( O(n^2 \log n) \).

**Incremental Construction:** Green and Sibson gave a very clean algorithm for constructing the Voronoi diagram in 1977 [34]. Their algorithm supposes the Voronoi diagram for \( k \) points is already constructed and then construct the new diagram after adding one more point. The algorithm spends \( O(n) \) time per point insertion and results in a total complexity of \( O(n^2) \). In spite of its quadratic complexity, this algorithm has been the most popular method of constructing the Voronoi diagram. Implementation details can be found in Field (1986) [30].

**Divide and Conquer:** The divide-and-conquer technique can also be used to construct Voronoi diagram. In 1975, Shamos and Hoey published a paper which introduced an algorithm to construct the diagram in \( O(n \log n) \). Although the algorithm is rather difficult to implement, it can be done with careful attention to data structures. Details can be found in Guibas and Stolfi (1985) [35].

**Fortune’s Algorithm:** To avoid the complexities of the divide-and-conquer programming, most of the implementations for constructing the Voronoi diagram used the \( O(n^2) \) incremental algorithm even though the performance is slow. In 1985, Fortune invented a clever plane-sweep algorithm. This algorithm is as simple as the
incremental algorithm, but has worst-case complexity of $O(n \log n)$ [31]. Plane-sweep algorithms pass a sweep line over the plane, leaving at any time the problem solved for the portion of the plane already swept, and unsolved for the portion not yet reached. A significant advantage of Fortune’s algorithm is that the storage needed at any one time is often much smaller than the size of the diagram when $n$ is large,

### 2.3.4 Introduction to Delaunay Triangulations

Triangulation is a fundamental problem in computational geometry. It breaks complicated geometric objects into simple geometric objects. The simplest geometric objects are triangles in two dimensions and tetrahedra in three. Classical applications of triangulation include finite element analysis and computer graphics.

The Delaunay triangulation is a special triangulation which maximizes the minimum angle over all triangulations of the set of points. Because of its nice properties, it has been widely used in a number of applications such as communication networks, surface interpolation and nearest neighbors in a graph. Figure 2.11 is the Delaunay triangulation of a point set in two dimensions.

First, we explain what a triangulation of a set of points is.

**Definition 2.3.4.1:** A maximal planar subdivision is a subdivision $D$ such that no edge connecting two vertices can be added to $D$ without destroying its planarity.

In other words, any edge that is not in $S$ intersects one of the existing edges.

**Definition 2.3.4.2:** Let $S$ be a set of points in the plane. A *triangulation* of $S$ is defined as a maximal planar subdivision whose vertex set is $S$.

Figure 2.12 shows two possible triangulations of a point set $S$ of 4 points.

**Definition 2.4.3.3:** Let $S$ be a set of points, its *Delaunay triangulation* is a triangulation of $S$ such that every triangle’s circumcircle contains no other point of $S$ inside it. We denote the edges of a Delaunay triangulation as *Delaunay edges*.

From the definition 2.4.3.3, we can tell the upper triangulation of Figure 2.12 is
not a Delaunay triangulation. The lower one is a Delaunay triangulation of the 4 points.

2.3.5 Properties of Delaunay Triangulation

We already know that: for every vertex \( v \) of the Voronoi diagram of \( S \), the circle \( C(v) \) contains no other point of \( S \). You can easily find the similarity between this property of Voronoi diagram and the definition of the Delaunay triangulation. Actually, the Delaunay triangulation and Voronoi diagram are dual structures, each contains the same “information” in some sense, but represented in a rather different form. When you have one of them, the other one can be easily constructed.

**Theorem 2.3.5.1:** Given a point set \( S \) of \( n \) points in \( R^2 \), two points \( p_i, p_j \) are connected by a Delaunay edge if and only if there exists a location \( v \) that has equal distance to \( p_i \) and \( p_j \) and \( v \) is closer to \( p_i, p_j \) than to any other point of \( S \).
The location $v$ in the theorem is on the common boundary of the Voronoi cells of $p_i$ and $p_j$. This theorem suggests that, if we connect any two points of $S$ by a straight line on the condition that the Voronoi cells of the points are neighboring to each other, we can have a triangulation of $S$; this triangulation is just the Delaunay triangulation of $S$, according to the following theorem.

**Theorem 2.3.5.2:** Given a point set $S$, its Delaunay triangulation is the straight-line dual of its Voronoi diagram.

The Figure 2.13 is an illustration of this property. The proof of this theorem can be found in Preparata and Shamos [55].

**Theorem 2.3.5.3:** Given a point set $S$ in $R_2$, the Delaunay triangulation is the triangulation of $S$ that maximizes the minimum angle among all triangulations of $S$.

In many applications where a triangulation is needed, triangles with a very slim shape and sharp angles are not preferred because of their computational instability and difficulty. In addition, Delaunay triangulation has the nice property of having
small vertex degree and small total length experimentally.

2.3.6 Incremental-Flip Algorithms for Delaunay Triangulation

The construction of Delaunay triangulation of a set of points is a popular topic in the area of geometric algorithms. Various different approaches have been studied in recent years.

The divide-and conquer method [24] for Delaunay triangulation recursively divides the point set $S$ into two subsets until each subset has only two or three points. Then the method can connect them directly. In the conquer step, subsets are merged and edges are flipped to meet the Delaunay criterion. Actually, the conquer step is equivalent to the construction of the dividing chain between the Voronoi Diagrams of two subsets of $S$ in the divide-and conquer method of the Voronoi diagram. Therefore, basically, these two methods are still the same method. An elegant and relatively new
method is the convex hull method [54].

An $n$-dimensional Delaunay triangulation can be achieved easily from a computation of the convex hull in $n+1$ dimensions. For example, a 2D Delaunay triangulation of a point set $S$ is equivalent to the projection of the convex hull of a 3D point set $S_t$ that is derived from a lifting transformation: $x_t = x; \ y_t = y; \ z_t = x^2 + y^2$.

Here we just introduce the incremental-flip algorithms which are relatively easy to be implemented.

The basic idea of the incremental-flip algorithm is the following. Let $S$ be a set of $n$ points in 3D. Let $4 < i \leq n$ and assume that the Delaunay triangulation of the first $i - 1$ points in $S$ is already obtained; Call it $D_{i-1}$. Add the $i$-th point $p_i$ to the triangulation, and correct the resulting triangulation to make it Delaunay again by flipping. The result is $D_i$. Repeat this process until $i = n$. This strategy always leads to the Delaunay triangulation of $S$ if implemented correctly.

The algorithm above is easy to implement if the points are sorted when the $p_i$ is added. $p_i$ lies always outside $D_{i-1}$. Sometimes, we want to add the points randomly, not in a sorted sequence. We can, first, assume an artificial “big enough” tetrahedron containing all points of $S$, and make it the initial triangulation. Therefore, all points in $S$ can be added inside the existing triangulation. Second, use a direct acyclic graph (called the history dag) to store the history of all performed flips. The resulting triangulation includes artificial triangles connecting each face of the convex hull of $S$ with one of the points at infinity. Those triangles are removed in a post processing step. In the worst case, the incremental-flip algorithm takes time and storage $O(n^2)$, where $n$ is the number of points in $S$.

2.4 Medial Axis

The concept of the medial axis of an object was introduced by Blum in [11] in 1967 for studying biological and medical shapes. Medial axis of an object is a skeleton-like
shape that provides a compact representation of its features and their connectivity. The word *axis* is originally used in two dimensions. For the sake of consistency, it is still used in three dimensions, e.g., medial axis surface.

Let $F$ be a smooth compact surface without boundary and $s$ be a point sample from $F$. A ball is called *medial* if it meets $F$ only tangentially in at least two points. The medial axis is defined as the closure of the set of centers of all medial balls. There are two medial balls at each point on $F$, one touching it from outside and the other one touching it from inside. Because the medial ball touches the surface tangentially, the line which goes through a point $s$ and the centers of its medial balls is normal to $F$ at $s$. For each point on the medial axis, if the distance to the boundary of the object is also stored, the resulting structure is known as the medial axis transformation (MAT) and the entire boundary representation can be reconstructed from it.

The MAT has been used in many fields. In modeling, for example, the medial axis or its associated radii can be used to deform the boundary of the object [58]. This natural deformation can be used in computer animation [61]. The uniform change of the radii along the entire medial axis can be used to generate the offset surface [7]. In path planning [36, 40], the medial axis serves as a guide because the path on it has a large distance from obstacles. In mesh generation [57], the medial axis provides a natural boundary to decompose the domain into sub-domains which are easy to mesh. MAT is also used in surface reconstruction, shape interrogation and simplification, injection molding simulation, feature recognition, etc. Actually, the medial axis is such a useful structure that it has been suggested by some researchers to use it as an alternate representation to B-rep in a geometric modeling system, or to the traditional boundary and constructive solid geometry (CSG) representations [20, 39].

The Figure 2.14 shows the medial axis of a rectangle and three medial balls touching the boundary from inside.
There are two significant drawbacks when the medial axis is used. First, small changes in the boundary of the object can lead to large changes in the medial axis. Second, it is difficult to compute because of the underlying algebraic complexity. For a polyhedron, the surfaces constituting the medial axis are quadrics, and the seam curves can have degree four. For solids with curved boundaries, the medial axis sheets and seam curves can have much higher degree. But because of its wide application, it has been studied by a lot of researchers. Many algorithms have been proposed to compute or approximate the medial axis. Culve, Keyser and Manocha [20] and Hoffman [39] have given algorithms for computing the exact medial axis for some special class of shapes. Tamal K. Dey and Wulue Zhao give an algorithm for approximating medial axis as a Voronoi Subcomplex [23]. Mark Foskey et al. provided an algorithm for efficient computation of a simplified medial axis, called $\theta$-simplified medial axis ($\theta$-SMA).

2.5 Poles

Let $D$ be the Delaunay triangulation of the set of points $S$, and $V$ be the Voronoi diagram, the dual of $D$. For a point $s$ in $S$, let $V_s$ be its Voronoi cell.

Amenta et al. [3] defined the poles as the following. For each point $s$ in $S$, if $s$ does not lie on $\text{conv}(S)$, the farthest Voronoi vertex $p^+$ in $V_s$ is called the positive
pole $p$. If $s$ does lie on $\text{conv}(S)$, let $p^+$ be a point at “infinite distance” outside the convex hull with the direction of $sp^+$ equal to the average of the outward normals of hull faces meeting at $s$. The negative pole of $s$ is the farthest point $p^-$ from $s$ such that two vectors from $s$ to $p^+$ and $p^-$ make an angle more than $\pi/2$. See Figure 2.15.

2.6 Simulation of simplicity

The data sampled from objects are rarely in general position. Edelsbrunner and Mücke developed the Simulation of Simplicity (SoS) method to slightly perturb the data so that the data are to be in general position [28]. Although the idea is simple, it is very difficult and to develop a perturbation that is small enough and does not change the non-degenerate position of points relative to one another. The SoS method performs the perturbation symbolically by replacing the parameters that represent the data by polynomials in $\epsilon$. SoS is like a black box that breaks ties between the degenerate simplices.

The SoS method has been implemented for arbitrary dimensions. But the data we
consider in this dissertation are in three dimensions. So let $S$ be a three-dimensional point set containing $n$ points. For each point in $S$, a unique index $i$ is assigned. Each point $p_i$ is represented by its three coordinates $(x_{i,1}, x_{i,2}, x_{i,3})$. The symbolic perturbation replaces each coordinate $x_{i,j}$, for $1 \leq i \leq n$ and $1 \leq j \leq 3$ by a polynomial in $\epsilon$:

$$x_{i,j}(\epsilon) = x_{i,j} + \epsilon(i, j)$$

The $x_{i,j}(\epsilon)$ is called perturbed or $\epsilon$-expanded coordinate. The polynomial $\epsilon(i, j)$ is called the $\epsilon$-expansion for the original coordinate $x_{i,j}$. The set of perturbed points is denoted as $S(\epsilon)$.

The choice of the polynomials $\epsilon(i, j)$ requires the following conditions to meet:

1. $S(\epsilon)$ must be simple.
2. $S(\epsilon)$ must retain all non-degenerate properties of the original set $S$.
3. $\epsilon(i, j) > 0$ but is arbitrarily small.
4. $\epsilon(i, j) \gg \epsilon(u, v)$ if and only if $(i, j) \prec (u, v)$.

The "\(\prec\)" denotes some linear order on the index pairs. The operator "\(\gg\)" means "arbitrarily much larger." The fourth condition means that coordinates with smaller index pairs are perturbed much more than those with larger index pairs.
CHAPTER 3

PREVIOUS WORK

Because of the widespread applications, the problem of surface reconstruction from unorganized points has been an active area of research for several decades. In this chapter, we will discuss some previous work on surface reconstruction.

3.1 Algorithms Related to Voxel Based Spatial Subdivision

Lorensen, W.E. and Cline, H.E. [50] provided an algorithm, called Marching Cubes, that creates a surface corresponding to a user-specified value from a set of 3D points. Marching cubes method subdivides space into a series of small cubes (voxels). For each vertex of a voxel, a value is assigned, then each vertex of the voxel is tested to be inside or outside the surface. The vertices with values greater than the user-specified value are inside the surface. The vertices with values less than the user-specified value of surface are outside the surface. A voxel cube intersects the surface if one or more vertices of it have values less than the user-specified threshold and one or more have values greater than this threshold. Once such a voxel is selected, the locations of the intersection points of the surface and the voxel can be calculated using interpolation. The algorithm processes a voxel and then marches to the next voxel. The surface is completed after all voxels are visited.

A voxel has eight vertices and each vertex can be either inside or outside the surface, there are $2^8 = 256$ possible ways the the surface passes through the voxel.
But the 256 cases can be reduced to 15 cases by using symmetry.

**Hoppe et al.** [42] described an algorithm that reconstructs an unknown surface from a set of unorganized points without knowing any structure of the data. The algorithm consists of two stages. In the first stage, they define a function $f$ that estimates the signed geometric distance to the unknown surface. The zero set $Z(f)$ is the estimate for the surface. In the second stage, the marching cubes method is used to approximate $Z(f)$.

In the first stage, an oriented tangent plane for each data point is computed to define a signed distance function. The tangent plane at every data point is represented as a center together with a unit normal vector. The center is computed as the centroid of the $k$-nearest neighbor points, and the normal vector is determined using principal component analysis (PCA) method [15]. If the surface is smooth and the data points are dense, the corresponding tangent planes are nearly parallel if two data points are close to each other. The normals are nearly in the same direction, otherwise, one of the two should be flipped. The idea of the method of orienting the planes is to first arbitrarily choose an orientation for some plane, then propagate the orientation from one point to another. The signed distance $f(p)$ from an arbitrary point $p$ to a unknown surface $M$ is the distance between the point and the closest point $z$ on $M$, multiplied by $\pm 1$, depending on which side of the surface $p$ lies. The oriented planes are used to mimic this procedure. First, they find the tangent plane whose centroid is closest to $p$, and then set the signed distance $f(p)$ to be the distance between $p$ and its projection point onto that tangent plane. Finally, Marching Cubes method is used to extract the surface corresponding to the zero set $Z(f)$.

**Roth and Wibowoo** [56] suggested a similar voxel based spatial subdivision algorithm as Hoppe’s but they calculate the distance function in a different way. For each voxel vertex, the signed distance is computed by taking the weighted average of the signed distances of every point in the eight neighboring voxels. They also use the
marching cubes algorithm to extract an initial triangular mesh from the zero set of the signed distance function.

First, the voxel size is automatically or manually set, and then the data points are assigned to the voxel cells into which they fall. For each data point, an “outer” normal vector is calculated by finding the closest two neighboring points in the voxel grid, and then using these points along with the original point to compute the normal. As in Hoppe’s algorithm, the normal orientation is also an important problem because marching cubes will produce wrong results if the normal points are in the wrong direction. This is because a signed distance is required but a normal in the wrong direction will produce the wrong sign. The normal orientation is determined as follows. Each voxel grid has six axis directions ($\pm x, \pm y, \pm z$). If we look from infinity down each axis into the voxel grid, those voxels that are visible must have their normals point towards the viewing direction. The normal direction is fixed for these visible points. Then the normal direction is propagated to those neighboring voxels whose normals are not fixed by this procedure.

Algorri and Schmitt [1] also provided an algorithm based on spatial subdivision. First, they subdivide the rectangular bounding box of the data set by a regular voxel grid. Then, the voxels which are occupied by at least one point of the data set are extracted. The outer quadrilaterals of the extracted voxels are treated as a first approximation of the surface. This is like the cuberille approach of volume visualization [38].

By diagonally splitting each quadrilateral into two triangles, the surface is converted into a triangular mesh. A depth-pass filter is used to smooth the cuberille artifacts. The filter assigns a new position to each vertex of triangle. This new position is computed as the weighted average of its old position and the position of its neighbors. The approximation of the resulting surface is improved by wrapping it towards the data points. How to identify an “outer” quadrilateral is the most impor-
tant step in this algorithm. Unless the voxels that contain the sample points are next to each other and bound a volume, “outer” cannot be defined. The disadvantage of the algorithm is that it is applicable only to data points that lie on fixed grid points, such as medical imaging data. The data points have to be sufficiently close to each other and the size of the voxel have to be appropriately chosen so that the set of voxels bound a volume.

3.2 Algorithms Related to Irregular Spatial Subdivision

In the last section, we have discussed some algorithms related to voxel based spatial subdivision. They have the same feature that the bounding box of the given data set is subdivided by a regular voxel grid. In this section, we will discuss some algorithms related to a basic irregular spatial subdivision: Delaunay triangulation.

Boissonnat, J. D. [14] starts with the Delaunay triangulation of the point set, then eliminates the tetrahedra that satisfy some rules from the boundary of the convex hull. Those tetrahedra with two faces, five edges and four points or one face, three edges and three points on the boundary of the current polyhedron are eliminated. Because of this rule, only objects without holes can be reconstructed. The tetrahedra are removed according to decreasing decision values. The decision value is the maximum distance of a face of the tetrahedron to its circumscribed sphere. The algorithm stops when all points lie on the boundary.

Isselhard, Brunnett, and Schreiber [43] improved the approach of Boissonnat [14]. The algorithm of Boissonnat can not handle objects with holes. Isselhard et al. modified the removal rule in the algorithm so that holes also can be constructed. The algorithm also starts with the Delaunay triangulation of the point set. The difference of the removal procedure is that more types of tetrahedron can be eliminated. The additionally allowed types of tetrahedra are those with one face and four points or three faces or all four faces on the current surface as long as no point would be-
come isolated through their elimination. The algorithm stops when all data points are already on the surface. If not, continue to remove more tetrahedra to recover sharp edges of the object. In this situation, the eliminations rules are restricted to those of Boissonnat, assuming that all holes present in the data have been already recovered. Additionally, the decision value of the tetrahedra is scaled by the radius of the circumscribe sphere as a measure for the size of the tetrahedron. In this way, the cost of small tetrahedra is increased which are more likely to be in regions of sharp edges than big ones.

Attali, D. [43] addressed an algorithm that also uses Delaunay triangulation for spatial decomposition. The solution of the algorithm is defined as a normalized mesh, which is included in the Delaunay triangulation. The normalized mesh consists of Delaunay elements (edges, faces and tetrahedra) whose dual Voronoi elements intersect the surface of the object. In two dimensions, given the sample point set $S$ taken from a curve $c$, the normalized mesh of $c$ consists of all edges in the Delaunay triangulation whose Voronoi dual intersects $c$. The normalized mesh retains all the topological properties of the curve if the curve belongs to a class of curves of bounded curvature, the so-called $r$-regular shapes. A bound on the sampling density can be given in the $r$-regular shapes. The curve $c$ is reconstructed by considering the angle between the intersections of the two possible circles around a Delaunay edge. The angle between the circles is defined to be the smaller of the two angles between the two tangent planes at one intersection point of the two circles. All edges whose associated Delaunay discs have an angle smaller than $\pi/2$ are chosen to be part of the reconstructed mesh. The idea is useful because the Delaunay discs tend to become tangent to the boundary of the object. If data points are sufficiently sampled, the reconstructed mesh is same as the normalized mesh, which is the same as $c$ topologically.

The normalized mesh is a correct reconstruction of shapes in two dimensions.
However, it cannot be extended to three dimensions. The reason is that some Delaunay spheres can intersect the surface but they are not approximately tangent to it in 3D. Therefore, in three dimensions, all faces of the surface are not contained in the normalized mesh. To deal with this problem, two heuristics are presented to complete the surface. The first heuristic is to triangulate each polygon of the border independently from the others. Given a simple polygon on the border of the surface, a triangulation of this polygon can be performed by adding one after another Delaunay triangles sharing two or three edges with this polygon. If the triangulation fails, the surface cannot be closed by this method. This approach has the advantage to make absolutely no assumption on the distribution of sample points. It deals with closed as well as open surfaces. Nevertheless, it may sometimes not provide the expected solution. The second heuristic is volume-based. Delaunay tetrahedra are merged to build up the possibly different solids represented in the point set. The set of mergeable solids is initialized with the Delaunay tetrahedra and the complement of the convex hull. The merging step is performed by processing the Delaunay triangles according to decreasing diameters. If the current triangle separates two different solids in the set of mergeable solids, they are merged if the following hold:

- no triangle from the normalized mesh is removed;
- merging will not isolate sample points inside the union of these objects, i.e. after the merge, the sample point must still belong to the boundary of at least one object.

Finally, the boundary of the resulting solids is considered as the reconstructed surface. The approach has the advantage that the skeleton of objects can easily be deduced. The skeleton is a very famous representation of objects from image analysis and computer vision. it is a thin figure which retains all the topological properties of the objects.
Bajaj, Bernardini and Xu [8] first calculate an approximate surface using *alpha*-shapes (*α*-shape is discussed in later sections) in a preprocessing phase and use the distance from the sample points to this surface as distance function. In the second step, the Delaunay triangulation is computed. By inspecting the signs of the distance function at the vertices, the tetrahedra traversed by the surface are found. For each of them, an approximation of the traversing surface is calculated. For this purpose, a Bernstein-Bézier trivariate implicit approximation is used. The approximation error to the given data points is calculated. A bad approximation leads to a further refinement of the tetrahedrization. The refinement is performed by incrementally inserting the centers of tetrahedra with high approximation error into the tetrahedrization. The process is iterated until a sufficient approximation is achieved.

The resulting surface is composed of trivariate implicit Bernstein-Bézier patches. A $C^1$ smoothing of the constructed surfaces is obtained by applying a Clough-Tocher subdivision scheme. Bernardini et al. [9, 10] extended and modified this algorithm by giving an additional mesh simplification step to reduce the complexity of the mesh represented by the *α*-solid. Additionally, the representation of sharp features can be achieved in the resulting surface.

3.3 Algorithm Related to Alpha Shapes

**Intuitive Definition of Alpha Shapes:** As mentioned in Edelsbrunner’s and Mücke’s paper [29], one can intuitively think of an alpha shape as the following. Imagine a huge mass of styrofoam making up the space and containing the point set $S$ as equal-sized pebbles. Using a sphere-shaped eraser we carve out all parts of the styrofoam we can reach without bumping into the pebbles inside. Thereby, we even carve out holes in the inside. That means we even carve out the parts not reachable by simply moving the eraser from the outside. Eventually, we will end up with an
object bounded by caps, arcs and points. If we now straighten all “round” faces to triangles and line segments, we have an intuitive description of what is called the alpha shape of \( S \). the alpha is the radius of the carving eraser. The Figure 3.1 is an example for this process in two dimensions.

![Figure 3.1: An example of an alpha shape of a set of points \( S \).](image)

In the process above, an eraser with very small radius \( \alpha \) will allow us to carve out all of the styrofoam except the pebbles inside. Thus the alpha shape of \( S \) degenerates to the point set \( S \) for \( \alpha \to 0 \). On the other hand, a very big value of \( \alpha \) will prevent us even from moving the eraser between two points since it’s too large. So we will never carve out the styrofoam in the inside of the convex hull of \( S \), and hence the alpha shape for \( \alpha \to \infty \) is the convex hull of \( S \).

Figure 3.2 shows a complete family of alpha shapes of a set of 4 points in two dimensions.

Figure 3.3 shows three alpha shapes of a set of points sampled from a head model. On the left is the original point set, corresponding to \( \alpha = 0 \). In the middle, an \( \alpha \) value is chosen for approximating the surface. On the right is the convex hull of the
Figure 3.2: A family of alpha shapes. Data set is from 3D Meshes Research Database by INRIA GAMMA Group.

Figure 3.3: Three alpha shapes.
point set, corresponding to $\alpha = \infty$.

Guo and Menon and Willette [37] described a method for reconstructing an unknown surface from a set of data points. The approach is to extract the surface as a polygon mesh from an alpha shape. Even though alpha shapes are generalized polytopes having complicated internal structures, they show that manifold surfaces, with or without boundaries, can be efficiently generated, and these surfaces completely describe the alpha shape to the extent that they are visible from outside.

A triangular mesh is simplicial if it is connected, orientable and each edge of the mesh belongs to exactly two faces, unless the edge is on the boundary of the mesh. Consider the decomposition of the complementary set of $S_\alpha$, into a collection of maximal connected components. The exterior set $E_\alpha$ of the alpha shape $S_\alpha$ is the unique unbounded component of this decomposition. An exterior face of $S_\alpha$ is a face that is on the boundaries of both $E_\alpha$ and $S_\alpha$. If the exterior faces form a simplicial surface, this surface is called an outer shell of $S_\alpha$. First, the entire family of alpha shapes for the data points is computed. Then they choose an alpha shape $S_\alpha$ which has an outer shell. Compute a directed normal for every faces so that the normals for the exterior faces point to the exterior set of the alpha shape. Using these normals, the algorithm enumerates the exterior faces to form an outer shell, which approximates the unknown surface.

Edelsbrunner H. [25] modified the original idea of Alpha Shapes method by assigning a weight to the points. The resulting weighted alpha shapes are a generalization of original alpha shapes. The original alpha shapes can be viewed as weighted alpha shapes with all weights equal to zero. The reason for assigning weights to the points is that the points are rarely uniformly sampled on the unknown surface. The density often varies with the curvature. If $\alpha$ is chosen so that a piecewise linear surface is produced in sparse regions, it will be clumsy and hide details in denser regions. Conversely, if $\alpha$ is chosen so that dense regions are nicely modeled then the
alpha shape will have holes and break apart in sparse regions. The assignment of large weights in sparse regions and of small weights in dense regions can be used to deal with this undesirable effect.

More details of implementation of weighted alpha shapes will be stated in the next chapter.

Teichmann and Capps [60] present the Anisotropic Density-Scaled Alpha Shapes method. The point density of each point is computed and it is used to get an approximation of the density of a triangle. For example, they average the densities of the triangle’s points or take the maximum of these densities. Then the $\alpha$-ball is reduced in size in areas where the triangles’ density is higher than average. By computing the density of the triangle in different ways (averaging, taking the maximum, and so on), one can achieve a finer level of detail in higher density areas, or detect neighboring objects if they have different densities. They also provided an approach to solve the interstice-problem in alpha shapes. The problem is that when the point density varies in different areas, larger $\alpha$ results in neighbored objects being connected resulting in bridges at sharp turns. The triangles spanning the bridges should be deleted from alpha shapes. They compress the $\alpha$-ball used for the alpha-test of the triangle in question along an axis perpendicular to a local plane separating the bridge. This procedure amounts to elongating space along that direction. To find the axis, the approximation to normal vectors is assumed to be available, e.g. the normal vectors of the real surface are available at every point.

In the classical alpha shapes, the triangle belongs to the shape since its circumscribed sphere has a smaller radius than $\alpha$. However in the anisotropic alpha shapes, the triangle is removed by modifying the $\alpha$-ball to be an ellipsoid whose axes and eccentricity are determined according to the local point normal information. The underlying assumption here is that the field of normals is sufficiently smooth.
3.4 Algorithms Related to Voronoi Diagrams

Amenta et al. [3, 5] give a simple combinatorial algorithm *Crust* that computes a piecewise-linear approximation of a smooth surface from a finite set of sample points. The idea is to use Voronoi vertices to remove triangles from the Delaunay triangulation. This algorithm extends their previous work [4] of curve reconstruction. In [4], they defined a planar graph on the sample points called the *crust*. The crust is the set of edges in the Delaunay triangulation of the sample points that can be enclosed by circles empty not only of sample points, but also Voronoi vertices. And they proved that if the sample points are sufficiently dense, the crust contains exactly the edges between sample points adjacent on the curve, i.e. the crust approximates the curve well. The crust is also a subset of the Delaunay triangulation of the sample points; adding the Voronoi vertices filters out the unwanted edges from the Delaunay triangulation. This technique is called *Voronoi filtering*.

But this idea cannot be directly extended to reconstruct a smooth two-dimensional manifold in three dimensions. The problem is that some vertices of the Voronoi diagram may be very close to the surfaces, thereby leaving holes in the crust. For example, the Voronoi center of a sliver can lie arbitrarily close to the surface. A *sliver* is a tetrahedron with bad aspect ratio yet a reasonably small circum-radius to shortest edge ratio, such as the tetrahedron formed by four nearly equally spaced vertices around the equator of a sphere. In [3, 5], the problem is solved by only considering the *poles* of the Voronoi diagram. First, they assume that the distance between sample points is proportional to the distance to the medial axis. Let $S$ be the set of sample points and let $P$ denote all positive poles $p^+$ and negative $p^-$, except those $p^+$ at infinite distance, the Delaunay triangulation of $S \cup P$ is computed. Keep only those triangles in which all three vertices are sample points. The resulting triangles form the crust. The crust is not necessarily a manifold. It often contains all four triangles of a sliver. Normal filtering and trimming are used to produce a
guaranteed piecewise linear manifold homeomorphic to the unknown surface. The limitation of this algorithm is that it is not reliable when the surface has boundaries.

Amenta, Choi, Dey and Leekha [6] introduced the *CoCone* algorithm which improved the Crust algorithm in theory and practice. The CoCone algorithm avoids the second Delaunay triangulation computation. It uses a concept called cocone for every sample point \( p \) to filter candidate triangles. The normal at each sample point on the surface \( S \) is estimated using *poles*. Let \( V_p \) be a Voronoi cell at a sample point and let the positive pole be the farthest Voronoi vertex of \( V_p \). The line through the sample point \( p \) and the positive pole is almost normal to \( S \) and is called the estimated normal line at the sample point. For an angle \( \theta \), they define a cone-complement - the cocone at \( p \) - as the complement of the double cone with apex \( p \) making an angle of \( \pi/2 - \theta \) with the axis that is aligned with the estimated normal line at \( p \). They determine the set of Voronoi edges in \( V_p \) that intersect the cocones at all three of the sample points inducing the edge. The dual triangles of these edges form the candidates triangles when \( \theta < \pi/8 \).

The CoCone and Crust algorithms have the following similar steps as following. First, a set of triangles are filtered from the Delaunay triangulation of the sample set. Second, all the sharp edges and their incident triangles are deleted in a recursive manner. This could lead to a cascaded deletion of the entire surface by deleting a triangle incident to a boundary vertex, so such triangles are not removed. Third, the resulting triangles lie flat to the surface but some edges could still have more than two triangles incident to them. These triangles usually form a double surface locally. A manifold can be extracted by walking on the inside or outside of the set of resulting triangles.

Amenta, Choi and Kolluri [7] described an algorithm called *Power Crust* which produces a surface mesh and an approximate medial axis from a set of points sampled from the surface of a three-dimensional object. The approach is to first
approximate the medial axis transform (MAT) of the object then use an inverse transform to produce the surface representation from the MAT.

The algorithm does not depend on the quality of the input point sample. It takes any input and gives an output surface which is “watertight” boundary of a three-dimensional polyhedral solid which is described by the approximate MAT. “Watertight” in the paper means the surface bounds a solid without holes. This unconditional guarantee makes the algorithm quite robust and eliminates the polygonalization, hole-filling or manifold extraction post processing steps required in previous surface reconstruction algorithms. The power diagram [26] is the weighted Voronoi diagram of a set of weighted points. The weights of points represent the radius of the spheres centered at those points. Combining these two concepts, the Power Crust method extracts the reconstructed surface as a power diagram of the largest empty spheres centered at the Voronoi vertices of the given set of sample points. Certain Voronoi vertices which are found to lie close to the surface are considered to be not on the medial axis and are removed. The major drawback of this algorithm is that it can be used only for surfaces with no boundaries.
In this chapter, we elaborate on the idea of alpha shapes. The definition of alpha shapes we present here is the one presented by Edelsbrunner and Mücke [29].

4.1 General Position

The input points of alpha shapes are assumed to be in general position. The following conditions must hold:

1. No four neighboring points lie in the common plane. This condition means that there are no degenerate simplices (flat tetrahedra, triangles with all three vertices collinear) that many be formed.

2. No five neighboring points lie on a common sphere. This condition insures that there are no ties; therefore, the Delaunay triangulation of $S$ is unique.

3. For any fixed $\alpha$, the smallest sphere through any 2, 3, or 4 points of $S$ has a radius different from $\alpha$. This condition insures that each alpha shape in the resulting family of shapes is unique.

In practice, input data rarely come in general position. However, there are methods for handling this inconvenience [28]. So we assume that the general position conditions are met.
4.2 Formal Definition of Alpha Shapes

In the last chapter, we have an intuitive definition of alpha shapes. There are several formal definitions of an alpha shape, $S_\alpha$, corresponding to a given three dimensional point set, $S \subset \mathbb{R}^3$, and a given value $\alpha$. The definition we present here is the one presented by Edelsbrunner and Mücke [29]. First we define some topological and domain specific terms.

**Definition 4.2.1:** If a set of points $T$ of size $|T| = k + 1$, is affinely independent, its convex hull, $conv(T)$, is a $k$-simplex, denoted by $\sigma_T$. $k = |T| - 1$ is the dimension of $\sigma_T$.

Thus a point is a 0-simplex, a line is a 1-simplex, a triangle is a 2-simplex and a tetrahedron is a 3-simplex.

**Definition 4.2.2:** A simplicial complex, $C$, is a collection of $k$-simplices, for $k \geq 0$, that satisfies the following two properties:

1. For every $T' \subseteq T$, if $\sigma_T \in C$ then $\sigma_{T'} \in C$. This means, for every simplex $\sigma_T$, $C$ contains all faces of $\sigma_T$ as well.

2. If $\sigma_T \in C$ and $\sigma_{T'} \in C$, then either $\sigma_T \cap \sigma_{T'} = \emptyset$ or $\sigma_T \cap \sigma_{T'} = \sigma_T \cap T' = conv(T \cap T')$. This means, the intersection of any two simplices in $C$ is either empty or a face of both simplices.

We call the highest dimension of any simplex $\sigma_T \in C$ the *dimension* of $C$. A subset $C' \subseteq C$ is a subcomplex of $C$ if it is also a simplicial complex. We call the union of all simplices of $C$ the underlying space of $C$, denoted by $|C|$. The underlying space of a simplicial complex $C$ is also called the polytope of $C$.

**Definition 4.2.3:** A simplex is $\alpha$-exposed if its vertices are on the bounding shell of an empty ball with radius $\alpha$. Formally, for $0 \leq k \leq 2$, let $\sigma_T$ be a $k$-simplex where $T \subseteq S$ and where the size of $T$ equals $k + 1$. $\sigma_T$ is $\alpha$-exposed if there is an empty $\alpha$-ball $b$ with $T = \partial b \cap S$. $\partial b$ is the bounding shell for the $\alpha$-ball $b$. 
**Definition 4.2.4:** Let $F_{k,\alpha}$ represent the set of all $\alpha$-exposed $k$-simplices for some fixed $\alpha$ and for some value of $k$ where $0 \leq k \leq 2$. Then $S_\alpha$, the alpha shape, is the polytope whose boundary consists of the triangles in $F_{2,\alpha}$, the edges in $F_{1,\alpha}$ and the vertices in $F_{0,\alpha}$. The $k$-simplices in $F_{k,\alpha}$ are also called the $k$-faces of $S_\alpha$.

### 4.2.1 Delaunay Triangulations

Different values of alpha define different shapes for the same point set $S$. Since we are assuming linear connections, the set of alpha shapes is finite. In their work [29], Edelsbrunner and Mücke showed that the entire discrete family of alpha shapes can be represented by the Delaunay triangulation, $D$.

From computational geometry, the Delaunay triangulation of a three-dimensional point set decomposes the convex hull of the point set into tetrahedra such that the circumsphere of each tetrahedron is empty. Using Edelsbrunner and Mücke’s notation, we defined the Delaunay triangulation as the set of $k$-simplices, $F_k$, for $0 \leq k \leq 3$, where for each simplex $\sigma_T$, the $k + 1$ vertices in the set $T \subseteq S$ lie in the shell that bounds an open empty ball (i.e., $\partial b \cap S = T$).

For each of the tetrahedra of $D$, no points of $S$ are associated with tetrahedron except for the points that are vertices of the tetrahedra. Each tetrahedron connects four closest neighbors. For each tetrahedron in $D$, there is some $\alpha$ that causes the tetrahedron’s faces, edges and vertices to be $\alpha$-exposed. From this, one can infer that Delaunay triangulation of a point set $S$ represents the entire family of alpha shapes. The formal representation is:

$$F_k = \cup F_{k,\alpha}, 0 \leq \alpha \leq \infty$$
4.3 Determining Alpha Intervals

We have already known in last section that the beauty of the alpha shapes idea is that entire family of shapes can be represented by an annotated version of the Delaunay triangulation. For each simplex in the Delaunay triangulation, it is sufficient to note a few intervals (of values for \( \alpha \)) which determine the status of that simplex. Once this annotated Delaunay triangulation is computed, the alpha shape corresponding to any particular \( \alpha \) can be produced quickly and easily.

**Definition 4.3.1:** Each \( k \)-simplex of \( \sigma_T \) of \( D \) defines an open ball \( b_T \) bounded by the smallest sphere \( \partial b_T \) that contains all points of \( T \). Let \( \varrho_T \) be the radius of \( b_T \).

For \( k = 3 \), \( \partial b_T \) is the circumsphere of \( \sigma_T \), for \( k = 2 \), the circumcircle of \( \sigma_T \) is a great circle of \( \partial b_T \), and for \( k = 1 \), the two points in \( T \) are antipodal on \( \partial b_T \). Call \( \partial b_T \) the smallest circumsphere and \( \varrho_T \) the radius of \( \sigma_T \). For \( 1 \leq k \leq 3 \) and \( 0 \leq \alpha \leq \infty \), define \( G_{k,\alpha} \) as the set of \( k \)-simplices \( \sigma_T \in D \) for which \( b_T \) is empty and \( \varrho_T < \alpha \).

Furthermore, define \( G_{0,\alpha} = S \), for all \( \alpha \). The sets \( G_{k,\alpha} \) do not necessarily define a simplicial complex because it can happen that \( G_{3,\alpha} \) contains a tetrahedron but not all triangles of this tetrahedron belong to \( G_{2,\alpha} \). It is similar for triangles and edges. With this in mind, we define the alpha-complex of \( S \), denoted by \( C_\alpha \), as the simplicial complex whose \( k \)-simplices are either in \( G_{k,\alpha} \) or they bound \((k + 1)\)-simplices of \( C_\alpha \).

Both the surface and the interior of an alpha-shape are subsets of the Delaunay triangulation. If a tetrahedron is determined to be in the alpha-complex, then the triangles, edges, and vertices binding it are also in the alpha-complex. Similar reasoning can be used for triangles and edges found in the alpha-complex. An interval can be defined for each simplex \( \sigma_T \) of the Delaunay triangulation such that \( \sigma_T \) is part of the alpha shape if and only if \( \alpha \) is part of the interval. Since the \( \varrho_T \) is the smallest circumsphere that bounds \( \sigma_T \), \( \sigma_T \) is \( \alpha \)-exposed for the interval \((\varrho_T, \infty)\).

Using the alpha-complex, the interval, \((\varrho_T, \infty)\), can be subdivided into three disjoint intervals: singular, regular, and interior. A simplex \( \sigma_T \) of the alpha-complex is considered
interior if the simplex is not part of the alpha shape, regular if it belongs to the alpha shape and bounds some higher-dimensional complex that is in the α-complex. For example, a triangle that is part of the surface of the α-complex is regular if it is a face on a tetrahedron that is part of the interior of the α-complex. \( \sigma_T \) is considered singular if it is part of the surface and does not bound any higher dimensional simplex in the interior of the α-complex.

Since a simplex in a Delaunay triangulation causes its lower dimensional components to be Delaunay, those components might not be singular because their smallest circumsphere encloses other points of \( S \). Those edges and triangles with this property are termed attached. Delaunay edges and triangles with smallest circumspheres not containing other points of \( P \) are considered unattached.

Recall that \( \varrho_T \) is the radius of the smallest circumsphere of \( \sigma_T \). In order to break up the interval for which \( \sigma_T \) belongs to \( C_\alpha \), Edelsbrunner and M"ucke introduce values \( \underline{\mu}_T \) and \( \overline{\mu}_T \) for which \( \sigma_T \) changes from singular to regular and from regular to interior, respectively. Before that, let \( up(\sigma_T) \) be the set of all simplices in \( D \) that contain a simplex \( \sigma_T \in D \), with \( |T| \leq 3 \), as a proper face, that is,

\[
up(\sigma_T) = \{ \sigma_{T'} \in D \mid T \subset T' \}
\]

If \( \sigma_T \) is a tetrahedron, define \( \underline{\mu}_T = \overline{\mu}_T = \varrho_T \). Otherwise,

\[
\underline{\mu}_T = \min \{ \varrho_{T'} \mid \sigma_{T'} \in up(\sigma_T), \text{ unattached} \}
\]

and

\[
\overline{\mu}_T = \max \{ \varrho_{T'} \mid \sigma_{T'} \in up(\sigma_T) \}
\]

It is sufficient to consider only the set

\[
up_1(\sigma_T) = \{ \sigma_{T'} \in up(\sigma_T) \mid |T'| = |T| + 1 \}
\]
that is, all faces incident to \( \sigma_T \) whose dimension is one higher than that of \( \sigma_T \), in order to derive the values \( \underline{\mu}_T \) and \( \overline{\mu}_T \): 

\[
\underline{\mu}_T = \min \left( \{ \rho_{T'} \mid \sigma_{T'} \in \text{up}_1(\sigma_T), \text{unattached} \} \cup \{ \mu_{T''} \mid \sigma_{T''} \in \text{up}_1(\sigma_T), \text{attached} \} \right)
\]

and 

\[
\overline{\mu}_T = \max \{ \mu_{T''} \mid \sigma_{T''} \in \text{up}_1(\sigma_T) \}
\]

The intervals of \( \alpha \) values in which \( \sigma_T \) is an interior, regular, or singular simplex of \( C_\alpha \) are shown in Table 4.1.

Here, we summarize the steps for computing the intervals. The Delaunay triangulation is computed first for the point set \( S \). Each simplex in the Delaunay triangulation is marked as either on or off the convex hull of the point set. Next, each simplex is classified as *attached* or *unattached*. A simplex \( \sigma_T \) is attached if and only if one of the simplices containing \( \sigma_T \) has a vertex within the smallest circumsphere of \( \sigma_T \). Next, the three alpha intervals for each complex are computed. The computation time for classifying each simplex is proportional to the number of simplices in the Delaunay triangulation because each simplex has only a constant number of faces. In other words, assuming that constant time suffices to decide whether or not a point belongs to \( b_T \), a simplex can be classified in constant amortized time.

### 4.4 Limitations of Alpha Shapes Method

Given a set of points, a family of alpha shapes can be produced. An alpha shape is expected to best approximate the surface of the point set. But there are two main problems here:

1. Sometimes there is no \( \alpha \) that exposes the surface of the point set. e.g. all alpha shapes don’t give good approximations of the surface.
Table 4.1: Alpha Intervals.

<table>
<thead>
<tr>
<th>$\sigma_T$ is .....</th>
<th>singular</th>
<th>regular</th>
<th>interior</th>
</tr>
</thead>
<tbody>
<tr>
<td>tetrahedron</td>
<td></td>
<td></td>
<td>$(\varrho_T, \infty)$</td>
</tr>
<tr>
<td>edge or triangle</td>
<td>not on convex hull(S), unattached</td>
<td>$(\varrho_T, \mu_T)$</td>
<td>$(\mu_T, \infty)$</td>
</tr>
<tr>
<td></td>
<td>not on convex hull(S), attached</td>
<td>$(\mu_T, \mu_T)$</td>
<td>$(\mu_T, \infty)$</td>
</tr>
<tr>
<td></td>
<td>on convex hull(S), unattached</td>
<td>$(\mu_T, \infty)$</td>
<td>$(\mu_T, \infty)$</td>
</tr>
<tr>
<td></td>
<td>on convex hull(P), attached</td>
<td>$(\mu_T, \infty)$</td>
<td>$(\mu_T, \infty)$</td>
</tr>
<tr>
<td>vertex</td>
<td>not on convex hull(S)</td>
<td>$(0, \mu_T)$</td>
<td>$(\mu_T, \infty)$</td>
</tr>
<tr>
<td></td>
<td>convex hull(S)</td>
<td>$(0, \mu_T)$</td>
<td>$(\mu_T, \infty)$</td>
</tr>
</tbody>
</table>

$up(\sigma_T)$ set of higher dimensional simplices in $\mathcal{D}$ containing $\sigma_T$

$\varrho_T$ radius of the circumsphere of the simplex $\sigma_T$.

$\mu_T$ the smallest radius of the circumspheres of the unattached simplices in $up(\sigma_T)$

$\overline{\mu}_T$ the largest radius of the circumspheres of the simplices in set $up(\sigma_T)$

2. How to find the alpha shape that best represents the surface?

The first problem often happens when the points are not uniformly sampled from the surface. The reason for the unsatisfactory result is that alpha shapes are based on the distances between points of the simplices in the Delaunay triangulation in order to decide if the simplex belongs to the shape or not. Points which are far apart as compared with $\alpha$ are not neighbors since the cutting ball will carve away the intervening space. Points which are close together form a barrier to the cutting ball and end up defining a patch of surface. For each choice of $\alpha$, there will be both areas where the details of the densely sampled parts will be hidden or undesired holes will appear in sparsely sampled areas. See Figure 4.1 and Figure 4.2.

In Figure 4.2, the left is a set of points that is generated using peaks function from Matlab. The 10,000 points are uniform in $xy$ coordinates, but not in $z$ coordinate.

The middle is the surface for $\alpha = 0.169$, holes are introduced to fulfill the no bridge criteria. The right is the surface for $\alpha = 0.278$. No holes, but bridges now exist between the peaks. Actually, there is not a value of $\alpha$ that produces a shape without holes and without bridges.
Figure 4.1: If the cutting ball is too big, the details are hidden from the surface. If the cutting ball is too small, holes are left on the surface.
Figure 4.2: Limitation of Alpha Shapes method. The data set is generated using Matlab.

The second problem assumes that there is $\alpha$ that exposes the surface, the difficulty here is how to find the $\alpha$. So far, we have an interactive slider where you can drag back and forth until the result looks good.

4.5 Weighted Alpha Shapes

Edelsbrunner [25] presents a modification that associates a weight with each point $(p, w_p)$: a high weight causes the simplices associated with that point to remain $\alpha$-exposed longer; a low weight causes simplices to disappear from the shape sooner. A
zero weight defaults to the unweighted alpha shape. See Figure 4.3.

In this section, we will discuss it with a formal definition.

**Definition 4.5.1:** A *weighted point* is denoted as $p = (p', p'')$, with $p'$ its location and $p''$ its weight. For two weighted points, $p = (p', p'')$ and $x = (x', x'')$, define

$$
\pi(p, x) = |p'x'| - p'' - x'',
$$

(4.1)

where $|p'x'|$ is the Euclidean distance between their locations. Here, we can treat $p$ as a sphere with center $p'$ and radius $\sqrt{p''}$. $x$ is treated as a sphere with center $x'$ and radius $\sqrt{x''}$. If both $p''$ and $x''$ are both positive then $\pi(p, x) = 0$ if and only if the two spheres intersect at an angle of 90 degrees.

Consider a subset $T \subseteq S$, a simplex $\sigma_T$ is considered $\alpha$-exposed if there exists a weighted point $p = (p', \alpha)$, so that

$$
\pi(p, x) = \begin{cases}
0 & \text{for all } p \in T, \text{ and} \\
> 0 & \text{for all } p \in S - T.
\end{cases}
$$

The weighted alpha shape of $S$, $W_\alpha = W_\alpha(S)$, is a polytope whose boundary is the union of all $\alpha$-exposed simplices spanned by subsets of $S$. These simplices are the faces of $W_\alpha$.

Consider an edge between two points. In the unweighted alpha shapes definition, the smallest circle that causes the edge to be $\alpha$-exposed is the circle whose diameter is the same as the length of the edge. When positive weights are associated with the points, the weights decrease the diameter of the smallest circle causing the edge to be $\alpha$-exposed. Negative weights have the opposite effect.

A modification of the original algorithm for computing unweighted alpha shapes is used to compute the weighted alpha shapes. First, the regular triangulation, which is analogous to the Delaunay triangulation, is generated. In the regular triangulation,
the point set is divided into tetrahedra such that a “weighted” circumsphere of each tetrahedron is empty. The same algorithm used to generate the Delaunay triangulation combined with the modified distance $\pi(p, x)$ in the equation can now be used to generate the regular triangulation.
As seen in Chapter 3, surface reconstruction methods have typically been design to exploit additional knowledge in specific problem instances. This chapter introduces the Gamma Shapes method, which makes few assumptions about the sample points and the unknown surface. No structure or other information is assumed within the point set $S$. The unknown surface may have arbitrary topological type.

5.1 Introduction

5.1.1 A Local Scale Factor: $\gamma$

In the last chapter, we talked about the limitations of alpha shapes. First, alpha shapes define a family of shapes. It is difficult to find out which alpha shape approximates the surface well. Second, the Alpha Shapes method can only be used to reconstruct the surface when the data points are uniformly sampled.

In this dissertation, we introduce Gamma Shapes in order to deal with the non-uniformity of a point set. We modulate the global factor $\alpha$ with a local scale factor, which we call $\gamma$, according to the density of the point set. As the cutting ball moves through space, the size of the ball is modulated by the local scale factor $\gamma$. When the point density is high in some area, a relatively small $\gamma$ is multiplied by $\alpha$ so that the details of the surface can be exposed. On the other hand, when the point density is low, a relatively big $\gamma$ is used so that unexpected holes on the surface can be avoided.
Figure 5.1: The radius of the cutting ball changes according to the point density. The \( \alpha \) is modulated by the value \( \gamma \).

(See Figure 5.1).

Let \( T \) be a subset of \( S \) of size \( |T| = k+1 \), with \( 0 \leq k \leq 3 \), and define \( V_T = \cap_{s \in T} V_s \). \( V_T \) is the set of all points for which there exists an empty open ball \( b_x \) centered at \( x \) with \( T \subseteq \partial b_x \cap S \). Equality holds if and only if \( x \) belongs to the relative interior of \( V_T \). Give a value of \( \alpha \), the simplex \( \sigma_T \) is \( \alpha \)-exposed if and only if there is a point \( x \) in the relative interior of \( V_T \) whose distance from the points in \( T \) is \( \alpha \).

We call balls of radius \( \alpha \) centered at points in \( S \) alpha balls. A simplex is \( \alpha \)-exposed if the Voronoi cell of its vertices has a common non-empty intersection with the set of alpha balls [16]; The resulting alpha complex of \( S \) is the Delaunay triangulation of \( S \) restricted to the alpha balls.

We consider the Delaunay triangulation restricted to a new set of balls whose radii are modulated by the \( \gamma \). Given each value \( \hat{\alpha} \in (0, \infty) \), we put a modulated alpha ball at each point \( s \in S \) with radius \( \alpha_s \), where

\[
\alpha_s(\hat{\alpha}) = \gamma \ast \hat{\alpha}
\]

Let \( G_s^{\hat{\alpha}} \) be the intersection of the Voronoi cell \( V_s \) and the ball at \( s \) and let \( G_s^{\alpha} \) be the interior of \( \cup_{s \in S} G_s^{\hat{\alpha}} \). The gamma shape is the Delaunay triangulation of \( S \) restricted
In our method, the $\gamma$ is evaluated at the points according to the local density. We specify the $\gamma$ for higher dimension simplex to be the minimum $\gamma$ of its components. Thus an edge has associated with it which is the minimum of the $\gamma$ at the two endpoints. A triangle is associated with a $\gamma$ which is the minimum of the $\gamma$ of its three vertices, and a tetrahedron is associated with a $\gamma$ which is the minimum of the $\gamma$ of its four vertices. For any simplex $\sigma \in D_S$, let $\alpha(\sigma)$ be the smallest $\alpha$ value at which $\sigma$ appears in the alpha complex and let $\hat{\alpha}(\sigma)$ be the smallest $\hat{\alpha}$ value at which $\sigma$ appears in the gamma shape. The $\hat{\alpha}(\sigma)$ can be computed from the value of $\alpha(\sigma)$.

$$
\hat{\alpha}(\sigma) = \max_{1 \leq i \leq k} \inf \{ \hat{\alpha} \mid \alpha_s(\hat{\alpha}) \geq \alpha(\sigma) \}
$$

The gamma shapes share many properties with alpha shapes. When $\hat{\alpha}$ is zero, the gamma shape consists just of the data points set $S$, and for sufficiently large $\hat{\alpha}$, the gamma shape is the convex hull of the data set. If $\hat{\alpha}_i < \hat{\alpha}_j$, the simplices in the gamma shape at $\hat{\alpha}_i$ are also in the gamma shape at $\hat{\alpha}_j$.

The process of assigning $\gamma$ to each simplex depends on the $\gamma(x, y, z)$ at the points. But where does it come from and how is it determined? Currently, we have two interesting candidates which have been used our work. They are Poles methods and Ellipsoid methods.

### 5.2 Geometric Definitions

One recent work on surface reconstruction from unorganized point sets is based on the idea of a crust [3, 5], in which Voronoi pole is the central part of the algorithm. We will make use of the Voronoi pole information in our gamma shapes.

The constructions of gamma shapes and the computation of poles are based on special properties of the Voronoi diagram of a set of points sampled from a surface.
Some sampling density assumption about the input sample data points is required to prove the theoretical guarantees about the quality of our methods.

5.2.1 Sampling Assumption

The assumption about the density of the data points set $S$ is taken from [3, 5].

**Definition 5.2.1.1:** Let $M_F$ be the medial axis of a surface $F$. The local feature size is a function $f : F \rightarrow \mathbb{R}$ where $f(s)$ is the distance to the medial axis from the point:

$$f(s) = \inf_{x \in M_F} \|s - x\|$$

The local feature size $f(s)$ measures the narrowness of surface $F$. $f(s)$ not only measures the local change of geometry near $s$, but also the relationship with other elements. The medial axis is close to the surface when the curvature is high or some other patch is nearby.

We should notice that the curvature and local feature size are two different concepts. Points $p$ and $q$, (for example, two points on an edge of a rectangle) on the surface may have the same curvature but their local feature sizes are different.

We assume the surface $F$ is a smooth surface. The distance from any point on $F$ to the medial axis is strictly greater than zero. If there is a sharp corner on $F$, the medial axis meets the surface. In such regions, the sampling density would have to be infinite.

**Definition 5.2.1.2:** A sample of data points $S$ of surface $F$ is an $\epsilon$-sample if every point $x \in F$ has at least one sample point $s \in S$ so that $\|x - s\| \leq \epsilon f(x)$.

The definition of $\epsilon$-sample means that each point on the surface $F$ has a sample point with $\epsilon$ factor of its local feature size. The smaller the $\epsilon$ is, the higher the density of $S$ is. Notice that the definition of $\epsilon$-sample is feature-dependent. It does not assume any global or even local uniformity. But it requires more sample points along smaller feature regions of the surface.
5.2.2 Poles

Consider the Voronoi diagram of a point cloud. Each edge is the bisector of the connection of a point to one of its neighbor points. Voronoi vertices are the intersections of these edges. The Voronoi cell associated with a point is defined by the polyhedron defined by the surrounding Voronoi vertices. When the local sampling is uniform and isotropic, the Voronoi cell is nearly spherical. When the local sampling is not uniform, the orientation of the cell and the relative distances to several Voronoi vertices from the point gives strong clues about the local behavior of the surface.

Figure 5.2: The poles of a site \( s \).

For a sample point \( s \), the poles \( p^+, p^- \) are two extreme Voronoi vertices on either side of the surface. See Figure 2.15. They are computed as follows [3, 5]:

1. If \( s \) does not lie on the convex hull, let \( p^+ \) be the farthest Voronoi vertex of \( V_s \) from \( s \). Let \( n^+ \) be the vector \( sp^+ \).

2. If \( s \) lies on the convex hull, \( p^+ \) does not exist. Let \( n^+ \) be the average normal of the triangles (incident on \( s \)) of the convex hull.
3. let $p^-$ be the Voronoi vertex of $V_s$ that is farthest from $s$ with negative projection on $n^+$. 

Where the sampling is dense, the Voronoi cell of every point $s$ is long and skinny and perpendicular to the surface because the Voronoi cell is bounded by the proximity of other samples on the same local patch of surface in directions tangent to the surface. The direction from $s$ to $p^+$ can be viewed as the approximation of the normal at $s$ on the surface. And the following description gives the clue of determining the $\gamma$ value:

1. $p^+$ (if it exists) is outside the surface (or perhaps far into the interior of the object). At least one of the sample points whose Voronoi cell shares this Voronoi vertex comes from a different object, or a distant, separate part of the same object.

2. $p^-$ (which always exists) is usually inside the surface.

3. The distance from the site to its associated negative pole, $p^-$, gives information about the local sampling density. If the point set is properly sampled, the density is proportional to the distance to the medial axis [5]. It is also known that the set of poles $s$ to the medial axis [7] as sample density approaches to infinity.

5.3 Local Pole Method

5.3.1 Theorem

The information contained in the poles gives a strong indication of local density of points. One way to determine the value of gamma is to use the distance from $s$ to $p^-$. That is

$$\gamma = \|s - p^-\|$$

We call this method Local Pole method.
Given the geometric definitions in last section, we can state the following theorem:

**Theorem 5.3.1.1:** Let $S$ be an $\varepsilon$-sample, $\varepsilon < 0.5$, for a smooth surface $F$. The gamma shape generated using Local Pole method with $\hat{\alpha} = \eta$ can be candidate triangles for reconstructing a 2-manifold homeomorphic to $F$.

### 5.3.2 Conditions for Homeomorphism

**Definition 5.3.2.1:** A function $\mu: X \rightarrow Y$ defines a *homeomorphism* between two compact Euclidean subspaces $X$ and $Y$ if $\mu$ is continuous, one-to-one and onto [6].

The Homeomorphism proof in [6] needs two conditions of the set of candidate triangles

1. The candidate triangles contain all triangles of the restricted Delaunay triangulation $D_S$.

2. The circumcircle of each triangle is small compared to the feature size at the triangle’s vertices.

Additional *normal filtering* and *manifold extraction* post-processing steps are required to generated the 2-manifold [5]. The gamma shapes provide only the candidate triangles for those post-processing steps.

### 5.3.3 Theoretical Guarantees

The following lemmas and proofs give the theoretical guarantees for the above theorem.

**Lemma 5.3.3.1:** [3] For any two points $p$ and $q$ on a surface $F$, $|f(p) - f(q)| \leq d(p, q)$.

**Proof:** $f(p) \geq f(q) - d(p, q)$, since the ball of radius $f(q)$ around $q$ contains the ball of radius $f(q) - d(p, q)$ around $p$ and contains no point of the medial axis. Similarly, $f(q) \geq f(p) - d(p, q)$.
Lemma 5.3.3.2: [3] Let $S$ be an $\varepsilon$-sample of surface $F$ and $s$ be a sample point in $S$. The intersection of $V_s$ and $F$ is contained in a ball of radius $\eta f(s)$, where $\eta = \frac{\varepsilon}{1-\varepsilon}$.

Proof: Let $x \in V_s \cap F$. Since $x$ is the closest sample point to $s$, using Lemma 1, we can get $d(x, s) \leq \varepsilon f(x) \leq \varepsilon (f(s) + d(x, s))$. So $d(x, s) \leq \eta f(s)$.

The following Lemma shows that the gamma shape generated using Local Pole method meets the first condition.

Lemma 5.3.3.3: Let $S$ be an $\varepsilon$-sample of surface $F$ and $s$ be a sample point in $S$. The gamma complex for $\hat{\alpha} \geq \varepsilon$ contains $D_s$.

Proof: Let $\sigma_1, \sigma_2, ..., \sigma_n$ denote the simplices incident on $s$ and let $\alpha_1, \alpha_2, ..., \alpha_n$ denote the smallest alpha value at which the simplices appear in the alpha shape. Let $\alpha_i$ be the largest of those alpha values. We have

$$\alpha_i \leq \varepsilon f(s)$$

On either side of $F$ at $s$, the center of the big tangent ball of radius $f(s)$ must lie within $V_s$ because $V_s$ contains all the points closest to $s$. Thus, the distance between $s$ and $p^-$

$$d(s, p^-) \geq f(s)$$

Therefore, using Lemma 2, we can get

$$\hat{\alpha}_i = \frac{\alpha_i}{d(s, p^-)} \leq \frac{\eta f(s)}{d(s, p^-)} \leq \eta$$

This completes the proof.

Lemma 5.3.3.4: For any point $x$ in the intersection of $V_s$ with the hyperplane containing $s$ and orthogonal to direction $s - p^-$,

$$\|x - s\| \leq \frac{\eta f(s)}{\sin \left(\frac{\pi}{2} - 3 \arcsin \eta \right)}$$
**Proof:** See [16].

**Lemma 5.3.3.5:** Let $S$ be an $\varepsilon$-sample of a surface $F$. The neighbors of $s \in S$ in a gamma shape for small values of $\hat{\alpha}$ are at distance at most

$$\left(\frac{1 + \hat{\alpha}}{1 - \hat{\alpha}}\right) \left(\frac{2\eta}{\sin(\frac{\pi}{2} - 3\arcsin \eta)}\right) f(s)$$

**Proof:** See [16].

This lemma states that the gamma shape is contained in a thickening of the surface $F$ where the thickness factor with respect to the local feature size depends on $\hat{\alpha}$ and $\eta$ via

$$\left(\frac{1 + \hat{\alpha}}{1 - \hat{\alpha}}\right) \left(\frac{2\eta}{\sin(\frac{\pi}{2} - 3\arcsin \eta)}\right)$$

The first part of the thickness factor only depends on the scale parameter $\hat{\alpha}$. The second part of the thickness factor only depends on $\eta$. We know $\eta = \frac{\varepsilon}{1 - \varepsilon}$. So the second part only depends on the sampling density $\varepsilon$. If $\hat{\alpha} = \eta$ and $\varepsilon < 0.1$, then $\varepsilon < 0.112$ and this factor is less than 1. That means the gamma shape does not contain any point from the medial axis of the surface. But in ordinary alpha shapes, given any $\varepsilon > 0$, we can find a surface such that the alpha shape contains a point of the medial axis.

The following Lemma shows that the gamma shape generated using Local Pole method meets the second condition.

**Lemma 5.3.3.6:** Let $S$ be an $\varepsilon$-sample of a surface $F$. All triangles incident to $s \in S$ in a gamma shape for $\hat{\alpha} < 1$ have circumradius of at most

$$\left(\frac{1 + \hat{\alpha}}{1 - \hat{\alpha}}\right) \left(\frac{2\eta}{\sin(\frac{\pi}{2} - 3\arcsin \eta)}\right) f(s)$$

**Proof:** See [16].

If $\hat{\alpha} = \eta = \frac{\varepsilon}{1 - \varepsilon} < 1$, then $\varepsilon < 0.5$. Now we can conclude that the gamma shapes
satisfy these two conditions of the set of candidate triangles. So, given a $\varepsilon$-sample set, $\varepsilon < 0.5$, the gamma shape generated using Local Pole method with $\hat{\alpha} = \eta$ can be candidate triangles for reconstructing a homeomorphic surface to the original surface $F$.

5.4 Global Pole Method

In the Local Pole method, we use only the $p^-$ from the Voronoi cell associated with the point. We know the set of poles converges to the medial axis [7] as sample density approaches to infinity. So we also consider all the $p^-$ from nearby Voronoi cells, and choose the nearest one. We call this method Global Pole method. See Figure 2.15.

From Figure 5.4, we can see that the Voronoi vertices associated with $s$ may be $p^-$ of nearby data points.

Figure 5.3 shows the modulated $\alpha$ balls by using $\gamma$. It is clear that both Local Pole and Global Pole methods reflect the correct density information of the data set.

Figure 5.5 shows that the local feature size of $s$ is not necessarily the same as the smallest radius $r$ of the medial balls touching $s$, that is $f(s) \leq r$.

The set of poles converges to the medial axis [7] as sample density approaches to infinity. The distance between a sample point $s$ and the nearest $p^-$ in its Voronoi cell is an approximation of the local feature size of $s$. When the sample density approaches to infinity, $d(s, p^-_{\text{global}}) = f(s)$. So the formula

$$\hat{\alpha}_i = \frac{\alpha_i}{d(s, p^-)} \leq \frac{\eta f(s)}{d(s, p^-)} \leq \eta$$

is still correct. This means all gamma shapes with $\hat{\alpha} \geq \eta$ contain all the triangles in the restricted Delaunay triangulation $D_S$. It implies that the gamma shape generated using Global Pole method still can be candidate triangles for reconstructing a surface homeomorphic to the original surface $F$. 
(a) Local Pole method

(b) Global Pole method.

Figure 5.3: Modulated $\alpha$ balls.
5.5 Ellipsoid Methods

This section describes two other methods to determine the value of $\gamma$. The basic idea is to fit an ellipsoid into a Voronoi cell and use the half length of the longest axis to be the value of $\gamma$. In practice, we let the centroid of the Voronoi cell to be the center of the ellipsoid and we also let the site in the Voronoi cell to be the center of the ellipsoid. We call these two methods $Ec$ method and $Es$ method.

In three dimensions, each Voronoi cell is a polyhedron surrounding a point $s$, except when $s$ is on the Convex Hull of the data set. We fit an ellipsoid to approximate
the Voronoi cell and the axes of the ellipsoid give information of position, size, and orientation in a similar way as we described in the Poles method.

In our implementation, we simplify the problem into fitting an ellipsoid to the set of vertices of the Voronoi cell using Principal Components Analysis (PCA) method [2, 15, 46].

Principal Components Analysis is a classical statistical method that has found applications in fields such as pattern recognition and image compression. The main use of PCA is to reduce the dimensionality of a data set and the resulting data set still contains most of the information in the original data set. For example, PCA can be used to fit a line to a set of points. The line gives the spread direction of the set of points.

First, we compute the covariance matrix of the Voronoi vertices of a Voronoi cell:

\[
V = \Sigma^k_{i=1} \begin{pmatrix}
X_i \cdot X_i & X_i \cdot Y_i & X_i \cdot Z_i \\
Y_i \cdot X_i & Y_i \cdot Y_i & Y_i \cdot Z_i \\
Z_i \cdot X_i & Z_i \cdot Y_i & Z_i \cdot Z_i
\end{pmatrix}
\]

where \( X_i = vx_i - \bar{vx} \), \( Y_i = vy_i - \bar{vy} \) and \( Z_i = vz_i - \bar{vz} \). \( vx_i, vy_i \) and \( vz_i \) are the coordinates of the Voronoi vertices, and \( \bar{vx}, \bar{vy} \) and \( \bar{vz} \) are the average of \( vx_i, vy_i \) and \( vz_i \), i.e. \((\bar{vx}, \bar{vy}, \bar{vz})\) is the centroid of the Voronoi cell.

Then we compute the unit-length eigenvectors \( V[0], V[1], \) and \( V[2] \) corresponding to eigenvalues \( e[0], e[1], e[2] \). There is a family of ellipsoids defined by them. For each Voronoi vertex \( p \), the ellipsoid is:

\[
E(p_i) = P_i^T \ast \left( \sum_{j=0}^{2} \frac{V[j] \ast V[j]^T}{L[j]^2} \right) \ast P_i = S
\]

where \( P_i = (X_i, Y_i, Z_i) \). The semi-lengths of the axes are \( L[j] = \frac{1}{\sqrt{e_j}} \). In our experiment, we choose the minimum volume enclosing ellipsoid by computing \( S \) to
be the maximum of $E(p_i)$, where $p_i$ is the Voronoi vertex.

If the point $s$ is on the convex hull of the set of data points, the $p^+$ does not exist and the Voronoi cell is not a closed polyhedron. In this case, we insert a $p_{new}^+$ to close the Voronoi cell. Even when the data point is not on the convex hull, it may be that $p^+$ is very far away. In this case, we replace $p^+$ by moving it closer to the centroid of the Voronoi cell.

$$p_{new}^+ = s + (p^+ - s) \times \frac{|p^- - s|}{|p^+ - s|}$$

Now, we have an ellipsoid that fits the Voronoi cell, we use the half length of the longest axis to be the value of $\gamma$.

From the steps above, we can see that the center of the ellipsoid is at the centroid of the Voronoi cell. In our experiment, we also tried to use the site in the Voronoi cell to be the center of the ellipsoid. That is, in the process of computing the covariance matrix, the coordinates of the data points $(x_i, y_i, z_i)$ is used instead of $(\bar{v}x, \bar{v}y, \bar{v}z)$

Figure 5.6 shows the fitted ellipsoid centered at the data points. Different sizes of ellipsoids reflect the density of the data points. We can see that it is consistent with Figure 5.3. Different shapes or orientations of the ellipsoids reflect the direction of the normals on the original surface.
Figure 5.6: The ellipsoids of body data set.
CHAPTER 6

RESULTS AND COMPARISON

In this chapter, we show some surfaces reconstructed by the Gamma Shapes method and compare them to the surfaces reconstructed by other previous methods. The results in this chapter are selected visually and we will describe how to select the shapes automatically. All the data sets are from the 3D Meshes Research Database (http://www-c.inria.fr/gamma).

6.1 Surfaces Reconstructed Using Gamma Shapes

Figure 6.1, Figure 6.2 and Figure 6.3 show some surfaces reconstructed using the Gamma Shapes method.

Figure 6.4 shows a rabbit data set of 16760 points.

From the descriptions of Local Pole and Global Pole methods, we know the value of $\gamma$ is smaller in the Global Pole method than in the Local Pole method, because the Local Pole method only considers the $p^-$ of the site itself. It means that the radius of the modulated alpha ball in the Global Pole method is slightly less than the radius in the Local Pole method. This means that the surface generated using the Global Pole method to contain more holes than using the Local Pole method. See Figure 6.5 and Figure 6.6.

Table 6.1 and shows the $\alpha$ values and triangles in the figures.

When the density of the data points changes dramatically, it is difficult to recon-
Figure 6.1: Surfaces generated using gamma shapes.

(a) Venus.

(b) Club.
Figure 6.2: Surfaces generated using gamma shapes.
Figure 6.3: Surfaces generated using gamma shapes.
Figure 6.4: A rabbit data set of 16760 points.

Figure 6.7 shows such a guitar case data set of 1080 points and the surface generated using gamma shapes.

6.2 Comparison to Alpha Shapes Method

For the problem of surface reconstruction from unorganized points, we have defined four methods of choosing $\gamma$. But which is the best one, and for a specific data set, which is more suitable for it? Here we compare them by giving some criteria of

<table>
<thead>
<tr>
<th>Rabbit</th>
<th>alpha value</th>
<th>triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Pole</td>
<td>0.929738</td>
<td>33785</td>
</tr>
<tr>
<td>Global Pole</td>
<td>0.988239</td>
<td>33776</td>
</tr>
</tbody>
</table>
Figure 6.5: Surface reconstructed using the Gamma Shapes method.

(a) Local Pole method.

(b) Global Pole method.
Figure 6.6: Surface reconstructed using the Gamma Shapes method.
Figure 6.7: Gamma Shapes method fails when density is low.
different methods. We have implementations of Alpha Shape, Gamma Shapes (Two Ellipsoid methods, Local and Global Pole method).

1. Find a parametric surface.

2. Generate data sets of sample points at various sampling rates. From the parametric function, we can generate normals for each point.

3. Run implementations on surface data sets.

4. Compare implementation to the original surface using the following metrics:

   (a) Average Angle of Deviation (ADA) for point normals
   (b) Highest Angle of Deviation (HAD) for point normals
   (c) Variance for point normals
   (d) Triangles
   (e) Triangle per point

For each sample point, compute an approximate normal from faces adjacent to point, then compute angle of deviation of approximate normal to actual normal given in point set. The average angle of deviation and highest angle of deviation and the variance are compared among the methods.

Figure 6.8 gives the reconstructed surfaces from the tori data. The inner torus has the same number of points (512) as the one outside. So the point density is higher on the inner torus, and lower on the one outside. The surface generated from alpha shapes contains a lot of bridges between the two tori. The Local Pole method and the Global Pole method reconstructed the exact triangles. Although the Ec and Es do not reconstruct the correct surfaces, they offer some insight on the automatic generation of the value of $\gamma$. They can provide a reasonable “educated guess” on which value of $\alpha$ produces the ideal shape.

Table 6.2 shows the properties of the surfaces generated by different methods.
Figure 6.8: The tori surfaces reconstructed using alpha shapes, Local Pole method, Global Pole method, Ec method and Es method.

<table>
<thead>
<tr>
<th></th>
<th>Alpha</th>
<th>Local</th>
<th>Global</th>
<th>Ec</th>
<th>Es</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADA</td>
<td>3.99</td>
<td>1.89</td>
<td>1.89</td>
<td>14.24</td>
<td>10.65</td>
</tr>
<tr>
<td>HAD</td>
<td>45.69</td>
<td>3.8</td>
<td>3.8</td>
<td>61.79</td>
<td>54.09</td>
</tr>
<tr>
<td>Variance</td>
<td>34.19</td>
<td>1.798</td>
<td>1.798</td>
<td>86.27</td>
<td>83.08</td>
</tr>
<tr>
<td>Triangles</td>
<td>2176</td>
<td>2048</td>
<td>2048</td>
<td>3045</td>
<td>2698</td>
</tr>
<tr>
<td>Triangles/Point</td>
<td>6.37</td>
<td>6</td>
<td>6</td>
<td>8.92</td>
<td>7.9</td>
</tr>
</tbody>
</table>

6.3 Comparison to Other People’s Work

In this section, we will compare our Gamma Shapes method to the Power Crust method and CoCone method.

In Figure 6.9, the upper surface is generated using Local Pole method and the bottom surface is generated using Power Crust method. In Chapter 3, we have described the Power Crust method. It generates a “watertight” surface. When the data is not a closed surface or you can’t tell which is the inside or the outside of the surface, the Crust method gives strange result 6.9(b). Our Gamma Shapes method
do not assume any other information on the data sets. The surface can be arbitrary shapes.

Figure 6.10 and Figure 6.11 show the surface generated from a contour data set of a bottle. From the pictures, we also can see that the surface generated using gamma shapes is better than the surface generated using CoCone methods and Power Crust.
Figure 6.9: Comparison of Local Pole method with Power Crust method.
Figure 6.10: Surface generated using gamma shapes.
Figure 6.11: Surface generated using CoCone and Power Crust methods.

(a) CoCone method.

(b) Power Crust methods
In this chapter, we will describe how we select the best shape automatically.

We have already known that both the Alpha Shapes and Gamma Shapes methods generate a family of shapes and both provide a global variable $0 \leq \alpha \leq \infty$ to select the corresponding shape. In the implementation, users can view the whole family of shapes by selecting the values of $\alpha$ through a slider bar. Users can visually select the “best” shape that reflects the original surface of the object as they expect.

In order to compare and analyze the Alpha Shapes and Gamma Shapes methods, the “best” shapes generated by them will be compared and analyzed. Sometimes, visually selecting the “best” shapes is not a good way to compare them because different users have different opinions. They may select different “best” shapes according to their expectations of the shape. In addition, this makes comparing and analyzing them very difficult and inaccurate.

So in our experiments, we provide a way to select the “best” shape automatically, instead of visually, based on the same criteria.

First, we define what is the “best” shape.

Although we don’t provide a theoretical analysis for the evaluation of the Alpha Shapes and Gamma Shapes methods, we provide a reasonable mathematical way to evaluate them. Details are described in the following subsections.
7.1 Select the best shape when original shape is available

This section describes how to select the “best” shape when the original shape is available for comparing.

As discussed in the last two chapters, both Alpha Shapes and Gamma Shapes methods generate a family of shapes from the data point set and each shape is defined by a global variable $0 \leq \alpha \leq \infty$. We call one reconstructed shape in the family to be “best” if it is closest to the original shape from which the data points were sampled.

First, we will describe how to measure the closeness between the original shape and one reconstructed shape.

We let

1. $O$ be the triangles in the original shape,
2. $R$ be the triangles in the reconstructed shape,
3. $O - R$ be the triangles in the original shape but not in the reconstructed shape,
4. $R - O$ be the triangles in the reconstructed shape but not in the original shape, and
5. $O \land R$ the triangles in both the original shape and the reconstructed shape.

Correspondingly, we let $|O|$, $|R|$, $|O - R|$, $|R - O|$ and $|O \land R|$ be the number of triangles of the shapes.

Figure 7.1 shows a case that there is one reconstructed shape matches the original shape perfectly.

We know the number of triangles in the original shape $|O|$ is a constant. So we normalized the number of triangles by dividing them by $|O|$, i.e, the percentages of $|O|$ instead of absolute numbers.

The “best” $\alpha$ is at
Figure 7.1: One reconstructed shape matches the original surface at some $\alpha$. 

(a) $|O - R|/|O|$. 

(b) $|R - O|/|O|$. 

(c) $|O \land R|/|O|$. 

Figure 7.1: One reconstructed shape matches the original surface at some $\alpha$. 
1. $|O - R|/|O|$ is a minimum,

2. $|R - O|/|O|$ is a local minimum, and

3. $|O \wedge R|/|O|$ is a maximum.

In this case, $|O - R|/|O| = 0$, $|R - O|/|O| = 0$ and $|O \wedge R|/|O| = 1$.

If all the reconstructed shapes in the family do not match the original shape, we let the “best” alpha be at

1. $|O - R|/|O|$ is a minimum,

2. $|R - O|/|O|$ is a local minimum, and

3. $|O \wedge R|/|O|$ is a maximum.

Figure 7.2 show such a case. We can easily find out where the “best” $\alpha$ is.

Let $|O \wedge R|/|O| = c$, we can get

1. $a = |O - R|/|O| = (|O| - |O \wedge R|)/|O| = 1 - c$

2. $b = |R - O|/|O| = |R|/|O| - |O \wedge R|/|O| = |R|/|O| - c$

3. $c = |O \wedge R|/|O|$

If one of the reconstructed surfaces is exactly the same as the original surface, we can see that the minimum of $a = 0$ and the maximum of $c = 1$ vote for a same alpha value. It is also easy to see that one of the local minimum values of $b = 0$ also votes for that alpha value.

If none of reconstructed surfaces matches the original surface, the minimum of $a$ and the maximum of $c$ still vote for a same alpha. So we can use the values of $a$ and $c$ to determine the “best” alpha value. The value of $b$ does not help much in this case because the local minimum values of $b$ do not have to be 0 in this case.
Figure 7.2: Dataset: Venus. The original surface is generated by using Clarkson’s Hull program. [19]
7.2 Selecting the best shape when original shape is not available

In the last section, we described a way that selects the “best” shape when the original shape is available to compare. In this section, we will describe a way to select a “best” shape or a shape close to the “best” shape when the original shape is not available.

As mentioned before, both the Alpha Shapes and Gamma Shapes generate a family of shapes from the data point set and each shape is defined by a global variable \( \alpha \). We call a shape \( S_\alpha \) and call the corresponding complex \( C_\alpha \) for a given \( \alpha \).

Recall the definitions of interior, regular and singular simplices:

A simplex \( \sigma_T \in C_\alpha \) is

1. **interior** if \( \sigma_T \) does not belong to the boundary of \( S_\alpha \),

2. **regular** if \( \sigma_T \) belongs to the boundary of \( S_\alpha \) and it bounds some higher dimensional complex in \( C_\alpha \), and

3. **singular** if \( \sigma_T \) belongs to the boundary of \( S_\alpha \) and it does not bound any higher dimensional complex in \( C_\alpha \).

First, we define two criteria for selecting the “best” shapes (Figure 7.3):

1. The corresponding \( C_\alpha \) does not contain any singular vertex.
2. The corresponding \( C_\alpha \) does not contain any singular edges.

These two criteria guarantee that each vertex in the complex belongs to at least one edge in the complex and each edge belongs to at least one triangle.

There may be more than one shape satisfy these two criteria, i.e., more than one \( \alpha \) value satisfy them. In out experiments, we define the shape to be “best” whose \( \alpha \) is smallest among those \( \alpha \) values that satisfy the two criteria.

Figure 7.4 and Figure 7.5 show the “best” shapes generated by the Alpha Shapes and Gamma Shapes methods based on the above two criteria.
Figure 7.3: Red: signature of singular vertices. Green: signature of singular edges.
Figure 7.4: "Best" shapes of hotdogs data set.
Figure 7.5: “Best” shapes of knot data set.
 CHAPTER 8

CONCLUSION AND FUTURE WORK

In this chapter we discuss the contributions of the work, and describe the future work we would like to work on.

8.1 Conclusion

In this dissertation, we first studied the preliminary concepts in the area of surface reconstruction. The surface reconstruction techniques are widely used in many application and many previous work has been done on this topic. We reviewed some popular methods and studied the Alpha Shapes method in details.

The main contribution of this dissertation may be summarized as follows. First and foremost, we described a surface reconstruction method to allow us to reconstructed a surface of a set of unorganized points in 3D. The method is called Gamma Shapes method. It extends the previous Alpha Shapes method presented by Edelsbrunner and Mücke [29]. The Alpha Shapes method completed similar tasks, but it was limited by the density of the data points. The Alpha Shapes method requires that the data points be uniformly sampled.

The reason that the Alpha Shapes method was limited to uniformly sampled data sets was because it relied on a global scale value \( \alpha \), which is used to determine the simplex in the Delaunay triangulation of the data points in the reconstructed surface or not. As described in the chapter 4, if the data points are not uniformly sampled
from the object’s surface, the surface may contain holes in the area where the points density is low and bridges in the area where the points density is high.

In order to deal with this problem, we make use of the medial axis, which can be used to indicate the point densities. In 2D, the Voronoi vertices converges to the medial axis when the sampling rates approaches infinity. In 3D, a set of special Voronoi vertices, called poles, converges to the medial axis when the sampling rates approaches infinity. In the Gamma Shapes method, the poles are used to compute a local scale factor $\gamma$ and the value of $\gamma$ is used to modulate the value of $\alpha$ in the Alpha Shapes method. Then the modulated $\alpha$ is used to determine if the simplex in the Delaunay triangulation belongs to the reconstructed surface. In this dissertation, we developed two ways to compute the value of $\gamma$: Local Pole method and Global Pole method. Two ellipsoid fitting methods are also used to compute the value of $\alpha$. Although these two ellipsoid fitting methods do not generate good surfaces, they do offer some insight on the automatic generation of the value of $\gamma$.

Another important contribution of our work is that we provide a way to select “best” reconstructed surface from the family of shapes generated by the Alpha Shapes method and Gamma Shapes method. Using the Alpha Shapes methods, users can view the whole family of surfaces by selecting the values of $\alpha$ through a slider bar. But they can only visually select the “best” reconstructed surface that reflects the original surface of the object as they expect. The disadvantage of visually selecting is that different users have different opinions, so they may select different “best” surfaces according to their expectations of the shape. It also makes comparing and analyzing the surfaces very difficult and inaccurate.

Finally, we showed the reconstructed surfaces generated by the Gamma Shapes and we compared them with the surfaces generated by other surface reconstruction methods such as Alpha Shapes method, CoCone and Power Crust method. We analyzed the surfaces generated by different methods by comparing the following prop-
erties of the surfaces:

1. Average Angle of Deviation (ADA) for point normals
2. Highest Angle of Deviation (HAD) for point normals
3. Variance for point normals
4. Triangles
5. Triangle per point

8.2 Future Work

The primary goal of this dissertation was to develop a method for constructing a surface from a set of unorganized data points. While we made great progress toward this end, there are still topics that must be analyzed and improved upon.

First, as described in the Chapter 5, we provided four ways to determine the value of $\gamma$: Local Pole, Global Pole, Ec and Es. We have compared our Gamma Shapes to some other surface reconstruction methods. So we could tell which reconstructed surface is the best based on its properties. In the future, we would like to work on this problem: given a set of unorganized points, which is the best choice among the four ways of determining the value of $\gamma$.

Second, we would like to modify the Gamma Shapes method to handle surfaces with sharp edges. The Voronoi cells of data points sampled from sharp edges are not long and thin. The poles computed from those Voronoi cells do not approximate the medial axis well. Our Gamma Shapes assumes the density of points is proportional to the distance between the data points and the medial axis. The sharp edges can be reconstructed when the sampled points is very dense. But in real data, this is often not the case.
At last, we would like to consider noisy data points in our reconstruction method. The data scanned from the laser range scanners are in the form of three dimensional points. However, the devices are not perfect devices. Measurement errors and noisy data points may be introduced in the process of scanning. If these problems are severe, it’s very difficult to reconstruct an accurate and consistent surface from the data points by only examining local regions of a point set independently [49]. Gamma Shapes uses a local scale factor $\gamma$ to modulate the $\alpha$ value so that it can not handle such errors and outliers.
Appendix A

IMPLEMENTATION

This chapter describes implementation details for the Gamma Shape method.

A.1 Triangle-Edge Data Structure

This section describes the triangle-edge data structure used by Edelsbrunner and Mücke to represent the Delaunay triangulation. [54].

As indicated by its name, the triangle-edge data structure is triangle based. Assume a unique index $f$ for each triangle in a triangulation $D$ of a set of points $S$ in three dimensions. The data structure’s basic unit is triangle-edge pair $a = < f, v >$, with $0 \leq v \leq 5$, which identifies one of six versions of triangle $f = \sigma_{p_i,p_j,p_k}$. Each version corresponds to one of the six directed edges incident to the triangle $f$.

It is convenient to associate edges or triangles with triangle-edge pairs. From a triangle-edge pair, the triangle and one of the triangle’s six directed edges can be determined. The six versions of triangle $f$ are shown in Figure A.1.

The positive halfspace of a triangle-edge $a$, denoted by $a^+$, is the set of all points $x$ which see the endpoints of $f_{i,j,k}$ make a counterclockwise turn. Each triangle $f$ defines two edge rings, one traversing the edges $f$ in counterclockwise direction, and one in clockwise direction. Each edge $e$ defines two triangle rings traversing the triangles incident to $e$ in the two opposite directions. Each triangle-edge $a$ belongs to exactly one edge ring and exactly one triangle ring.
The following basic triangle-edge functions are defined:

**a.org = i**:  
if $p_i$ is the origin (tail) of the directed edge $e_a$.  

**a.sym = b**:  
if $e_a = e_{i,j}$ and $e_b = e_{j,i}$, that is, $e_a$ and $e_b$ are the two directed versions of the same undirected edge.  

**a.enext = b**:  
if $e_a$ and $e_b$ are both in the same edge ring and $e_b$ is the successor of $e_a$.  

**a.fnnext = b**:  
if $e_a$ and $e_b$ are in the same triangle ring and $e_b$ is the successor $e_a$.  

Mücke developed an encoding scheme for each triangle-edge pair. A triangle-edge pare $(f, v)$ can be encoded

$$Encode(f, v) = 8 * f + v$$  

(A.1)
Mücke used eight instead of six as the coefficient of $f$ so that he could use the faster shift operator instead of multiplication. The decoding routing is shown as follows:

\[ f = \text{index div 8} \quad (A.2) \]

\[ v = \text{index mod 8} \quad (A.3) \]

All of the triangles in the Delaunay triangulation $D$ are stored in a dynamic array of triangle pairs.

The internal representation of the triangle-edge functions takes advantage of the fact that the structure of all edge rings is identical. Therefore, it can be implemented by a global lookup table $ve[0..5]$ with six entries $ve[v]$, $0 \leq v \leq 5$, such that, $a.enext = \langle t, ve[v] \rangle$, for any triangle-edge pair $a = \langle t, v \rangle$. However, actual pointers are needed for the successor relations in the triangle rings. Note that there are six versions of a triangle but only three vertices (that is, possible origins). To save memory, a second lookup table $vo[0..5]$ is used to map the indices corresponding to the origins of the six different versions of a triangle $f$ to its base triangle $\langle f, 0 \rangle$. See Figure A.2 for the internal declaration of the data structure and the pseudocode of the basic triangle-edge functions.

There are nine fields per record for a triangle $f$ in total: three origin fields for the indices, $i$, $j$, and $k$, where $\langle f, 0 \rangle \equiv f_{i,j,k}$, and six $next_f$ fields for the pointers to the next triangle in the triangle rings of $\langle f, v \rangle$, for $0 \leq v \leq 5$. For convenience, assume a (dynamically allocated) linear array $tr[0..\ast]$ to store all the triangle records.
A.2 Alpha Shape File and Data Structure

As mentioned in the previous chapter, we can represent the whole family of alpha shapes of \( S \) by the faces or simplices of the Delaunay triangulation \( D \), augmented by a set of intervals. The overall layout of the data structure is shown in Figure A.2. The detail is discussed below.

Since it is often convenient to have the interior of a shape triangulated, Mücke represents \( \alpha \)-complexes, rather than \( \alpha \)-shapes. As already mentioned, there is a single interval for each simplex \( \sigma_T \in D \) so that \( \sigma_T \) belongs to the \( \alpha \)-complex \( C_\alpha \) if and only if \( \alpha \) is contained in this interval. Each face is classified as either interior, regular or singular, which breaks down the interval for each face into three (possibly empty) subintervals; see Table 4.1.

A.2.1 Face Indexing

The data structure, which is used to store the Delaunay triangulation \( D \), is triangle based. The vertices, edges, and tetrahedra are only represented implicitly. However, some sort of indexing for faces of all dimensions is needed in order to conveniently store additional information for each face (that is, the above mentioned intervals) in linear arrays. The vertices and triangles are uniquely indexed. Triangles are indexed as in the triangle-edge data structure.

Mücke developed a hashing scheme to index edges and tetrahedra. For each edge and tetrahedron, the hash function provides a unique index between 1 and \( m_e > n_e \) and between 1 and \( m_t > n_t \), respectively, where \( n_e \) is the maximum number of edges and \( n_t \) is the maximum number of tetrahedra. There will be some unused indices because there are empty records in the hash tables. The number of unused indices is determined by the given load factor (for example, the load factor may be set to \( \frac{n}{m} = 0.8 \)).

Let \( E_{\sigma_T} \) be the set of all edge-triangle pairs that correspond to a simplex \( \sigma_T \in D \).
Figure A.2: The $\alpha$-shape file and data structure
For a tetrahedron, $E_{\sigma_T}$ consists of the four edge-triangle pairs that represent the tetrahedron's four faces. For an edge, $E_{\sigma_T}$ contains all the edge-triangle pairs of the facets in the two triangle rings (in opposite directions) corresponding to the edge.

Next, a minimum edge-triangle pair is defined for an edge or a tetrahedron:

$$a_{\text{min}} = \min\{< f, v > | < f, v > \in E_{\sigma_T}\} \quad (A.4)$$

The $a_{\text{min}}$ can be computed in constant time for each tetrahedron and in constant amortized time for each edge in $D$. $a_{\text{min}}$ can be used as a key for the functions because it is unique for each face. Moreover, the inverse indexing can be implemented. Given a face index $i$, the edge-triangle pair representing the face can be obtained by the $a_{\text{min}}$ reference in the $(i-1)$-st entry of the corresponding hash table, or, in the case of triangles, by directly referring to the $i$-th triangle record. Thus, a unique (and reversible) indexing scheme for all faces of $D$ is achieved.

A.2.2 Alpha Ranks

From the previous chapter, we already know that the set of $\alpha$-thresholds contains all end points of the intervals in Table 4.1. The sorted sequence of $\alpha$-thresholds, $\alpha_1, \alpha_2, ..., \alpha_r$, with $\alpha_r < \alpha_{r+1}$ for $1 \leq r \leq r$, forms the $\alpha$-spectrum. For convenience, we assume $\alpha_1 = 0$ and $\alpha_r = \infty$; if 0 and $\infty$ do not occur in the set of $\alpha$-thresholds, these values are added to the spectrum. We say the threshold $\alpha_r$ is of rank $r$. Each rank stands for an end point of a certain value. This is used to store the intervals of the faces of $D$ in four rank tables, one for each dimension. For edges and triangles we need to store the ranks of the $\varrho$, $\mu$, $\overline{\mu}$ values, for vertices the ranks of $\underline{\mu}$ and $\overline{\mu}$, and for tetrahedra only the rank of $\varrho$.

Note that certain intervals are empty. A “void” rank 0 is used to encode these.

- Attached edges and triangles have $\varrho - \text{rank} = 0$. 
• Faces bounding $\text{conv}(S)$ have $\mu - \text{rank} \neq 0$ and $\overline{\mu} - \text{rank} \neq 0$.

• Redundant (or dumped) vertices and empty entries in the edge and tetrahedra rank tables are marked with $\mu - \text{rank} = \overline{\mu} - \text{rank} = 0$.

A.2.3 Alpha Shape Files

Master List File: It is desirable to be able to retrieve, given a rank $r$, all faces whose intervals have $\alpha_r$ as an end point. For each $\alpha$-threshold $\alpha_r$ in the spectrum, Edelsbrunner and Mücke store the list of faces whose intervals have this threshold value as an end point. These lists are merged into a master list of sublists in order of increasing rank. Each entry of the master list stores the corresponding face index, the dimension of the face, the rank type (the entry may have type $\rho$, $\mu$, $\overline{\mu}$, and a flag indicating the end of a sublist. The faces in each sublist are sorted in order of non-decreasing dimension.

# Master list.
# Data: "venus"
# Ranks: 8025 (entries: 37472)
Rank [0], threshold = -3.402823e+38 (-infinity) ... 0 entries
Rank [1], threshold = 0.0 ... 711 entries
  v 711 [rho]
  v 710 [rho]
  v 709 [rho]
  v 708 [rho]
  v 707 [rho]
  v 706 [rho]
  v 705 [rho]
  ...
v 4 [\rho]  
v 3 [\rho]  
v 2 [\rho]  
v 1 [\rho]  
Rank [2], threshold = 6.806065e-03 ... 3 entries  
v 68 [\mu_1] v 61 [\mu_1] e 68 61 [\rho]  
Rank [3], threshold = 7.910558e-03 ... 3 entries  
v 439 [\mu_1]  
v 421 [\mu_1]  
e 421 439 [\rho]  
Rank [4], threshold = 8.622494e-03 ... 3 entries  
v 149 [\mu_1]  
v 148 [\mu_1]  
e 148 149 [\rho]  
...  
Rank [487], threshold = 2.911480e-02 ... 2 entries  
e 145 177 [\mu_1]  
f 145 159 177 [\rho]  
Rank [488], threshold = 2.912787e-02 ... 3 entries  
e 546 512 [\mu_1]  
e 512 553 [\mu_1]  
f 512 546 553 [\rho]  
Rank [489], threshold = 2.913403e-02 ... 2 entries  
v 15 [\mu_1]  
e 15 19 [\rho]  
...  
Rank [8023], threshold = 1.483023e+02 ... 7 entries
The master list is implemented as a linear array. The spectrum stores an index for each threshold $\alpha_r$ pointing to the corresponding sublist within the array. If we scan the spectrum and master list in order to decrease ranks, we get faces in the same order as they are removed from the $\alpha$-complex when “shrink warping” the $\alpha$-shape from the convex hull, for $\alpha_r = \infty$, to the point set $S$ itself, for $\alpha_1 = 0$. When the points are in general position, we note that the sublists of $\alpha_1$ and $\alpha_r = \infty$ are empty. We also can observe that it is not necessary to store the $\rho = 0$ values of the vertices. For $\alpha = 0$, the set of vertices is initialized with $S$, by default.
Appendix B

COMPUTATIONS IN GAMMA SHAPES

B.1 Computing Poles

One of the main tasks in poles methods is to compute the poles of each points. The set of poles is a subset of the Voronoi vertices. So first, we need to compute the Voronoi vertices of the data set. From the introduction of Voronoi diagram, we know that each Voronoi vertex is the center of the circumsphere of a tetrahedron in the Delaunay triangulation.

If we treat a point as a vector from the origin point to its position, let \( \vec{a}, \vec{b}, \vec{c}, \vec{d} \) be the four vertices of the tetrahedron, and let \( \vec{m} \) be the center of the circumsphere of the tetrahedron.

Let \( ax, ay, az \) be the components of \( a \), and likewise for \( b, c, \) and \( d \). Let \( |a| \) denote the norm of \( a \), and let \( a \times b \) denote the cross product of \( a \) and \( b \).

Let

\[
\vec{u} = |\vec{d} - \vec{a}|^2 \cdot [(\vec{b} - \vec{a}) \times (\vec{c} - \vec{a})]
\]

\[
\vec{v} = |\vec{c} - \vec{a}|^2 \cdot [(\vec{d} - \vec{a}) \times (\vec{b} - \vec{a})]
\]

\[
\vec{w} = |\vec{b} - \vec{a}|^2 \cdot [(\vec{c} - \vec{a}) \times (\vec{d} - \vec{a})]
\]

and let
\[ W = 2 \begin{vmatrix} bx - ax & by - ay & bz - az \\ cx - ax & cy - ay & cz - az \\ dx - ax & dy - ay & dz - az \end{vmatrix} \]

The center of the circumsphere is

\[ \vec{m} = \frac{\vec{u} + \vec{v} + \vec{w}}{W} \]

In the process of computing the Voronoi vertices, it is easy to remember the Voronoi vertices of a specific site because the center of the circumsphere of the tetrahedron is a Voronoi vertex of all the four vertices or site.

After this, we need to compute the positive pole \( p^+ \), which is the furthest Voronoi vertex from the site. If the site if not on the convex hull of the data points, it is very easy to find the \( p^+ \). But when the site is on the convex hull, the \( p^+ \) does not exist. In other words, it is in the infinity. If this is the case, we compute the convex hull of the data set and find all the triangles incident on the site. Then set the direction of \( p^+ \) to be the direction of the average normal of these triangles and set the length to be a big value.

Once we have the \( p^+ \), the next step is to find the \( p^- \), which is the furthest Voronoi vertex in the opposite direction of \( p^- \). That means

\[ \cos \theta = \frac{s\vec{p}^+ \cdot s\vec{p}^-}{\|s\vec{p}^+\|\|s\vec{p}^-\|} < 0 \]

or

\[ \theta = \arccos \frac{s\vec{p}^+ \cdot s\vec{p}^-}{\|s\vec{p}^+\|\|s\vec{p}^-\|} > \frac{\pi}{2} \]
B.2 Timing and Complexity

There are 3 steps to compute the gamma shapes. First, compute the alpha shapes (simplex intervals are also computed in this step). Second, compute the value of $\gamma$. Third, given an $\alpha$, modulate it by $\gamma$ and test the triangle is in the resulting shapes by comparing it to the intervals. The main work in the first step is to compute an annotated Delaunay triangulation. The complexity is $O(n \log n)$. The complexity of the second step is $O(n)$ because you just need to find the poles for each data point. The complexity of the third step is also $O(n)$ because it is related to the number of triangles in the Delaunay triangulation. Overall, the complexity is $O(n \log n)$.

In the first step, we use Ernst Mücke’s Alpha Shapes software. The second and third steps are implemented in C++. The running times, measured on a Dell Precision 380 desktop PC, given in table B.1 are the running time of second and third steps.

<table>
<thead>
<tr>
<th>objects</th>
<th>points</th>
<th>Local pole</th>
<th>Global pole</th>
</tr>
</thead>
<tbody>
<tr>
<td>tori</td>
<td>1024</td>
<td>0.40</td>
<td>0.41</td>
</tr>
<tr>
<td>triceratops</td>
<td>2832</td>
<td>0.99</td>
<td>1.04</td>
</tr>
<tr>
<td>body</td>
<td>2838</td>
<td>0.99</td>
<td>1.05</td>
</tr>
<tr>
<td>holes</td>
<td>4000</td>
<td>1.32</td>
<td>1.33</td>
</tr>
<tr>
<td>face</td>
<td>6737</td>
<td>2.21</td>
<td>2.16</td>
</tr>
<tr>
<td>knee</td>
<td>9535</td>
<td>3.05</td>
<td>2.99</td>
</tr>
<tr>
<td>peaks</td>
<td>10000</td>
<td>3.66</td>
<td>3.63</td>
</tr>
<tr>
<td>knot</td>
<td>10000</td>
<td>4.20</td>
<td>4.24</td>
</tr>
<tr>
<td>skull</td>
<td>10002</td>
<td>3.03</td>
<td>2.99</td>
</tr>
<tr>
<td>rabbit</td>
<td>16760</td>
<td>5.29</td>
<td>5.28</td>
</tr>
<tr>
<td>club</td>
<td>16864</td>
<td>5.59</td>
<td>5.64</td>
</tr>
</tbody>
</table>

Figure B.1 shows the plot of running time behavior of the Local Pole method.

B.3 User Interface

We also developed a visualization software, see Figure B.2, to display the surface generated using gamma shapes. It is implemented using Qt/QGL under Linux.
The visualization tool allows the user to interactively select $\alpha$-values and display the corresponding shape. With mouse, you can rotate, translate and scale the shape. It can save the current status of rotation, translation and scale information and load it later. You also can change the material color of the surface, turn on/off the data points and print the triangles out to a file.
Figure B.1: Running time of experimental data, in seconds
Figure B.2: Visualization tool.
Bibliography


[36] Leonidas J. Guibas, Christopher Holleman, and Lydia E. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based


