8-2001

The mathematics of surface reconstruction

Tamara S. Bouma

To the Graduate Council:

I am submitting herewith a thesis written by Tamara S. Bouma entitled "The mathematics of surface reconstruction." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Conrad Plaut, Major Professor

We have read this thesis and recommend its acceptance:

Ken Stephenson, David Anderson

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Tamara S. Bouma entitled "The Mathematics of Surface Reconstruction." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Conrad Plaut, Major Professor

We have read this thesis
and recommend its acceptance:

Ken Stephenson

David Anderson

Accepted for the Council:

Interim Vice Provost and
Dean of The Graduate School

The Mathematics of Surface Reconstruction

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Tamara S. Bouma

August 2001

DEDICATION

I would like to dedicate this thesis to my Husband Stephen.

His love and support has given me the confidence to pursue my dreams.

His confidence has made me believe these dreams are possible.

ACKNOWLEDGMENT

I would like to acknowledge the Engineering Department at the University of Tennessee, Knoxville. Specifically, I would like to thank David Page for the time and effort he gave in helping me understand some of the nuances of his field.

ABSTRACT

This thesis discusses mathematics engineers use to produce computerized three dimensional images of surfaces. It is self-contained in that all background information is included. As a result, mathematicians who know very little about the technology involved in three dimensional imaging should be able to understand the topics herein, and engineers with no differential geometry background will be able to understand the mathematics.

The purpose of this thesis is to unify and understand the notation commonly used by engineers, understand their terminology, and appreciate the difficulties faced by engineers in their pursuits. It is also intended to bridge the gap between mathematics and engineering.

This paper proceeds as follows. Chapter one introduces the topic and provides a brief overview of this thesis. Chapter two provides background information on technology and differential geometry. Chapter three discusses various methods by which normal vectors are estimated. In Chapter four, we discuss methods by which curvature is estimated. In Chapter six, we put it all together to recreate the surface. Finally, in chapter seven, we conclude with a discussion of future research. Each chapter concludes with a comparison of the methods discussed. The study of these reconstruction algorithms originated from various engineering papers on surface reconstruction. The background information was gathered from a thesis and various differential geometry texts.

The challege arises in the nature of the data with which we work. The surface must be recreated based on a set of discrete points. However, the study of surfaces is one of differential geometry which assumes differentiabile functions representing the surface. Since we only have a discrete set of points, methods to overcome this shortcoming must be developed. Two categories of surface reconstruction have been developed to overcome this shortcoming.

The first category estimates the data by data by smooth functions. The second reconstructs the surface using the discrete data directly. We found that various aspects of surface reconstruction are very reliable, while others are only marginally so. We found that methods recreating the surface from discrete data directly produce very similar results suggesting that some underlying facts about surfaces represented by discrete information may be influencing the results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

$\mathbb{R}$	The set of Real Numbers
$\mathbb{R}^3$	The set of Real Numbers in 3 Dimensional Space
Σ	Represents the original surface to be recreated
S	Represents the recreated surface
PC	Represents the collection of points created by the laser camera
n	Normally represents the number of points in the PC
p	Represents a particular point in the PC
p_i	Represents the i^{th} point in the PC
$\mathbf{N}_p, \mathbf{N}_i$	Represents the normal vector at p or at p_i
$\mathbf{T}_p, \mathbf{T}_i, \mathbf{t}_p, \mathbf{t}_i$	Represents the tangent vector to S or Σ at p or p_i
$\mathbf{A}, \mathbf{B}, \mathbf{K}$	Upper-case bold letters, except $\mathbf{N}$ or $\mathbf{T}$, represent general matrices
$\mathbf{x}, \mathbf{u}, \mathbf{v}$	Lower-case bold letters represent general vectors
λ_{i_j}	Represents eigenvalues
$k_{\min_p}, k_{\max_p}$	Represents Principal Curvatures, $k_{\min_p} \leq k_{\max_p}$
$\mathbf{e}_{i_k}$	Represents the eigenvectors corresponding to λ_{i_j} or principal curvatures
k_{n_p}, k_{n_i}	Represents the Normal Curvatures at point p or p_i
K_p, K_i	Represents the Gaussian Curvature at point p or p_i
H_p, H_i	Represents the Normal Curvature at point p or p_i

1 INTRODUCTION

Over the course of human history, theoretical mathematicians have developed some amazing methods by which to describe and quantify the world around us. Early engineers and scientists developed and used mathematics to create the technological and scientific advances of each age. It was used to determine when the rainy season would arrive, describe our solar system and how the planets interact, erect monuments, build the earliest plumbing systems, and countless other discoveries. These pioneers had to do the work of mathematicians, scientists,and engineers all at once. However, as mathematics, the sciences, and engineering became more complicated and as more was learned, they became separate pursuits and people started specializing on one area or the other.

Engineers and scientists study theoretical mathematics developed by mathematicians to develop their theories and technologies. Unfortunately, theoretical mathematics cannot usually be applied directly to the particular application they are developing. As a result, engineers and scientists are challenged to either alter the mathematical theory to their application or alter their application to fit mathematical theory. Mathematicians are often unaware of these challenges, and this thesis is an exploration into one area where these challenges have arisen.

In recent years, engineers have produced technology written about only in science fiction stories as recently as 40 years ago. Computer, space, and entertainment technologies have been built on the foundation of theoretical mathematics. Three dimensional imaging is another area of promising research which will eventually allow human beings to explore places currently inaccessible. It will allow us to take vacations to exotic locations and interact with the environment without leaving the comfort of our living rooms. It will allow robots to navigate their environment without human intervention, and allow them to recognize objects and make judgements on those objects. The challenge arises in how to use mathematical theory to create these images. The most obvious place to begin looking for mathematical solutions is Differential Geometry because it contains a study of surfaces in three-dimensional space. The challenge arises in the nature of the data we work with. As pictured in figure 1, we obtain an unorganized set of points representing various portions of the surface obtained via a range camera. From this set of points, our goal is to create an accurate computerized image of our original surface.

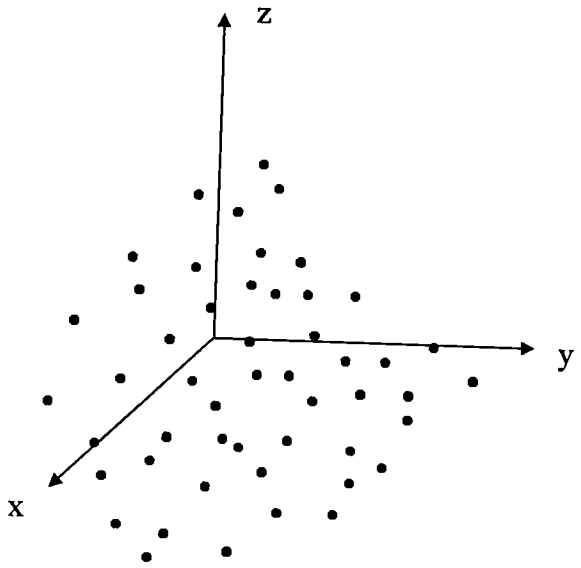

Figure 1: Illustration of a Point Cloud

We will be studying several different surface reconstruction algorithms developed over recent years. Our purpose is to explore the mathematical implementation of these algorithms, unify the notation between these methods, and, where possible, determine the mathematical viability of their arguments. In most of the methods described, the mathematical arguments are incomplete and not well organized. Attempts are made to complete these arguments and organize them into a logical format. The arguments are only completed if the intentions of the original author are well-known. No attempt is made to expand on an idea which is not started by the author. For example, if a calculation for curvature is given for the sphere, but none is given for an ellipsoid even though an ellipsoid is mentioned in the paper, this author will attempt to justify the calculation for the sphere, but not for an ellipsoid.

This thesis is organized as follows:

- Chapter 2 will consist of a discussion about the technology used to obtain data and the terminology commonly associated with this technology. We will also discuss some errors

inherent in this technology. Finally, we will give a brief overview of differential geometry consisting of commonly used definitions and theorems.

- Chapter 3 consists of a discussion about Normal Vector estimation. We begin by discussing methods which estimate normal vectors on smooth functions which approximate the data. Next, we will discuss methods which estimate normal vectors based on the discrete data directly.

- Chapter 4 focuses on various curvature estimations. In this chapter we will begin with a discussion about smooth function approximations. We will then discuss curvature estimation directly on the discrete data.

- Chapter 5 finally puts it all together and focuses on actual surface recreation. The first method focuses on the methods which initially fit smooth functions to the data. The last methods focus on the methods which use the data directly to recreate the surface.

1.1 Notation

Throughout this thesis, we will use the following notation. $\mathbb{R}$ will represent the real numbers and $\mathbb{R}^3$ will represent three dimensional space. Let Σ represent the original object we want to recreate, and S will represent the computerized image. Our set of sample points will be denoted by PC and each individual point in the PC will be denoted by p_i where $1 \leq i \leq n$ and n is the number of data points making up our set of sample points. Thus, $p_i \in PC$ is the i^{th} point in the point cloud. Let upper case bold letters such as $\mathbf{A}$ represent matrices, and let λ_i and $\mathbf{e}_i$ be the eigenvalues and corresponding eigenvectors for the matrix, $1 \leq i \leq m$ with m the number of eigenvalues, normally $m = 3$. Other bold lowercase letters, such as $\mathbf{v}$, will denote general vectors. The only exception to this case will be the notation for the normal vector; the normal vector will be denoted as $\mathbf{N}$. Thus, the normal vector at point p_i on some surface will be denoted $\mathbf{N}_{p_i}$. Finally, $k_{\min_{p_i}}$ and $k_{\max_{p_i}}$ will denote the minimum and maximum principal curvatures at point p_i on some surface; k_{n_i} will denote the normal curvature at point p_i on a surface, and K_{p_i} and H_{p_i} denotes the Gaussian and Mean curvatures respectively at point p_i on the surface.

2 THE FIRST STEP

2.1 Surface Representation

Suppose we have a surface of some object Σ that we want to recreate as a computer image, call it S. The first objective is to find a way to represent Σ in a way that the computer can interpret. We accomplish this with a range camera, and the specifics about the camera can be found in Chase [CS99]. The range camera will give us a set of N points representing the surface which will be fed into the computer. We will use these points to create S. Each point is represented by $p_i = (x_i, y_i, z_i)$, $1 \leq i \leq n$. Here, x_i represents the horizontal distance from the center of the camera; y_i is the vertical distance from the center of the camera; and z_i is the distance from the camera, or more easily stated, the depth from the center of the camera.

Definition 1 *A **point cloud**, denoted by PC, is the set of data points $\{p_i\}$ $1 \leq i \leq N$ created from a laser camera used to create S from Σ.*

Definition 2 *The finite set of points $\{p_i\} = PC$ **discretizes** the surface Σ. In other words, PC is a **discrete** representation of Σ.*

We begin by discussing the amount of information, or size of PC, the camera can create. Similar to computer monitors, the cameras work in terms of *pixels.* Computer monitors are classified in terms of pixels per square inch, or PPSI, cameras are classified the same way, and they range from 128 X 128 to 1024 X 2048. We can calibrate the camera to create the point cloud to any number of pixels limited only by its PPSI. In three dimensional imaging, a pixel is also referred to as a *voxel.* While a pixel is thought of generally as a flat square of a certain length and width, a voxel is a cube. Thus, a voxel is simply a pixel with depth equal to the length of its size.

Because we are creating a 3D image from a 3D surface, it is impossible to capture the entire object with a single pass of the camera; therefore, it is usually necessary to take several passes with the camera in different positions to capture obstructed portions of the object. Once we have taken enough passes with the camera to capture the entire object, a registration algorithm is necessary to merge these point clouds into a single set of data. The chosen algorithm should have a method to

locate points in each separate point cloud that represent the same portions of the object. This will allow us to match those points so in the merged point cloud, we can avoid redundancies.

Over the years, the technology used to create these point clouds has improved immensely. There are two primary methods by which cameras create the point cloud, sound and light. Each method has primary drawbacks, and when we consider which kind of camera to use, we must consider these drawbacks. There are two categories, accuracy and resolution. Accuracy refers to the "correctness" of the information, while resolution refers to how "sharp" the information is.

Definition 3 *We refer to* ***accuracy*** *as the correctness of the information received. In the context of range cameras, accuracy refers to the correctness of the measurement of the coordinates; specifically, the z-coordinate.*

Definition 4 *We refer to* ***resolution*** *as the sharpness of the information. In the context of range cameras, resolution refers to the size of each point captured on the surface. The smaller the point, the higher the resolution, the larger the point, the lower the resolution.*

Before we discuss the difference between accuracy and resolution, we first want to clarify what we mean by a "point in the point cloud," and what that point represents. Mathematically, it is impossible to collect points on the surface because points are infinitesimally small; therefore, the elements in the point cloud are not points as known to mathematicians; they would more accurately be described as pixels. Now we can discuss the difference between accuracy and resolution in the context of range cameras. A range camera is accurate if its measurements of the coordinates are nearly correct. A camera with low resolution means that the pixels are larger, which results in the image being blurry with some loss of detail. A camera with high resolution, on the other hand, means that the pixels are very small. Therefore, the resultant image is sharper and not as much detail is lost.

Remark 5 *Throughout the rest of this thesis, we will refer to elements in the point cloud as* ***points*** *to remain consistent with the name "Point Cloud."*

We can now begin discussing two different kinds of range cameras. Some models of cameras emit an ultrasonic "chirp" to determine the point cloud. Specifically, to create each element in the

point cloud, the camera emits a high-pitched chirp at the surface and calculates the time it takes the sound to travel back to the camera. Thus, for some $p_i = (x_i, y_i, z_i) \in PC$, x_i is determined by the horizontal distance the camera moves from center to capture the p_i; y_i is determined by the vertical distance the camera moves from center to capture p_i, and z_i is determined by the formula

$$z_i = \frac{v\triangle t}{2}$$

v is the speed of sound

$\triangle t$ is the time it has taken for the sound to travel back to the camera.

The sound cameras are accurate because of the low velocity of sound. However, the resolution is low because sound waves spread out as they leaves their source. As such, by the time it hits the surface, it covers a larger portion of the surface so the pixels are larger. If users want to eliminate overlap, they must calibrate the camera so that at each "chirp" covers a unique part of the surface. The problem with this is that the resulting point cloud is very sparse which yields unreliable calculations leading to an inaccurate surface reconstruction. If we allow the sound waves to overlap, we obtain points in the point cloud that overlap resulting in inaccurate calculations and a blurry image.

Other cameras, such as those currently used by the University of Tennessee, Knoxville, use a light laser to create the point cloud. Like the ultrasonic cameras, the x_i, and y_i for $p_i = (x_i, y_i, z_i) \in PC$, are determined in the same manner. z_i is determined by

$$z_i = \frac{c\triangle t}{2}$$

c is the speed of light

$\triangle t$ is the time it has taken the light to travel back to the camera.

These cameras produce a high resolution image since the laser can be concentrated into a very narrow beam; however, the information may be less accurate because of the high speed of light. Because the distance in relatively short, any small error in measuring $\triangle t$ will result in a relatively large error in the z_i coordinate. However, because technology is so advanced, the laser light cameras are becoming more and more accurate, and are the cameras of choice because of the high resolution

that is possible. In either case, determining the exact values x_i and y_i for each $p_i \in PC$ is specific to each camera and will not be discussed further. Of course, in both cases, the particular definition assumes that the camera moves vertically and horizontally. If the camera is stationary and simply rotates, the calculations would be converted to polar coordinates.

2.2 Errors

There are two different errors that can occur in the data set, or point cloud. One is inherent in the surface and technology, the other is a random error.

Definition 6 *We call* ***noise*** *the set of points in the point cloud that result from errors in measurement beyond a certain threshold.*

Definition 7 *We call* ***outliers*** *the set of points randomly created which are not indicative of a part of a surface.*

To illustrate the difference between outliers and noise, consider the following two examples. Suppose the camera has an error in measurment of ε. This means that the camera should measure the range of the object within ε of the true distance. Points that have an error $\varepsilon_1 > \varepsilon$ because of the surface color or type, are called *noise,* see figure 2.

Noise can be created by any number of things, from the light interfering with itself, to small rough regions on the surface on a microscopic level. However, we cannot do anything to accommodate these problems, and we might want those small rough spots in our recreation. These problems are irritating but are not the most common causes of noise in our data. The most common cause of noise is the surface itself. Shiny and very dark surfaces will cause noise. Shiny surfaces are reflective. A highly reflective surface, such as a mirrored surface, will reflect the light in the direction of the surface. As such, if the surface is highly reflective, the light from the camera will not be returned to the camera. Although our surfaces are usually not mirrored, noise may still occur. Very dark surfaces may absorb some of the light before reflecting it back. Both of these situations may cause errors in measurements, but are somewhat easy to accomodate by painting the surfaces gray if possible.

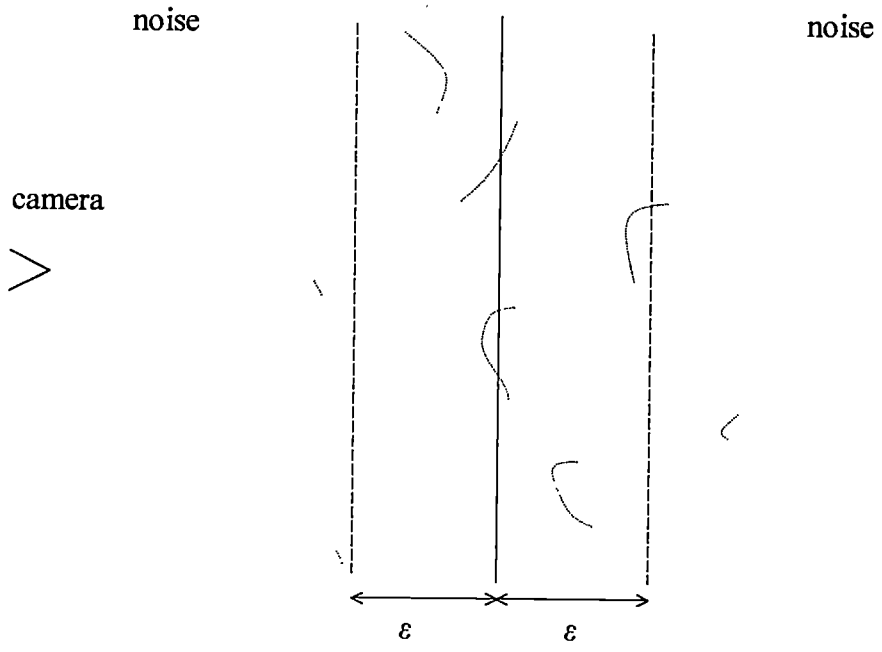

Figure 2: Illustration of Noise

For our example of outliers, suppose there is a bee in the room when we are using the laser camera. Furthermore, suppose the bee flies directly into the laser while the camera is working. Clearly, the bee is not part of the surface, yet a data point will be created as part of the PC. This is called an outlier, see figure 3.

Although these errors have different causes, they can both be handled the same way by simply ignoring those points. We have to determine how to decide whether an element in the point cloud is an outlier or noise. With a few well-placed assumptions about the surface, we can determine which points in the PC are outliers or noise. First, we can assume the surfaces are connected in some way. If we obtain a group of points in the point cloud which seem to indicate levitation, we consider those points to be outliers or noise. Another assumption we make is that the surfaces are relatively smooth. By this, we mean that the surfaces contain no abrupt changes. For example, we would not attempt to recreate a surface which has sharp spikes on it, or deep crevices.

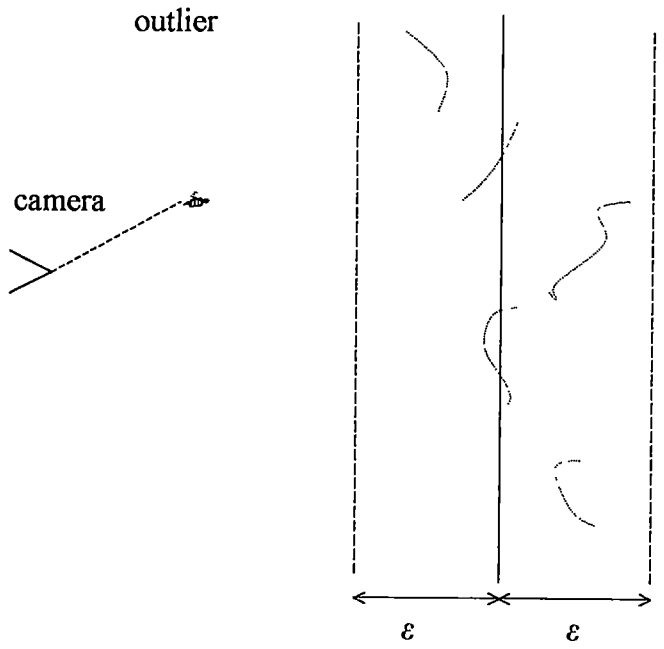

Figure 3: Illustration of an Outlier

We also assume that the camera is accurate more often than inaccurate. In both cases, we would eliminate elements in the point cloud which are some predetermined distance from its nearest neighbors. Finally, if we know something about the surface itself we can locate outliers and noise. We can estimate the distance the surface is from the camera and use this information to locate outliers and noise. Points which lie outside a predetermined distance threshold would be considered as noise or outliers. Furthermore, we can locate the noise and outliers if we know the depth of the object using a similar thresholding technique. Of course any algorithm designed to locate outliers and noise must incorporate any acceptable error generated by the camera technology itself.

2.3 Basic Mathematical Tools

Now that we have the point cloud, we must determine how we can interpret the data set. To guide us, we use differential geometry. Differential geometry is the study of curves and surfaces in some

kind of space. Applied to object recreation, we will look at properties of curves and surfaces in 3-space, or $\mathbb{R}^3$. Unfortunately, Differential Geometry cannot be applied directly to our data because Differential Geometry is a study of *differentiable,* or at the least, *piecewise differentiable* curves and surfaces. Nevertheless, we will define basic ideas from Differential Geometry drawn from DoCarmo [DC76].

Definition 8 *$\Sigma \subset \mathbb{R}^3$ is a **regular surface** if for each $p \in \Sigma$, there is a neighborhood $V \subset \mathbb{R}^3$ containing p and a map $f : U \to V \cap \Sigma$ of a connected open set $U \subset \mathbb{R}^2$ onto $V \cap \Sigma \subset \mathbb{R}^3$ such that*

1. *$f(u,v) = (x(u,v), y(u,v), z(u,v))$ where $(u,v) \in U$. The functions $x(u,v)$, $y(u,v)$, $z(u,v)$ have continuous partial derivatives of all orders in U.*
2. *f has an inverse $f^{-1} : V \cap \Sigma \to U$ which is continuous*
3. *For each $q \in U$, the differential $df_q : \mathbb{R}^2 \to \mathbb{R}^3$ is one-to-one. f is called a parametrization of surface Σ.*

Definition 9 *Let $g : V \subset \Sigma \to \mathbb{R}$ be a function defined in an open subset V of a regular surface Σ. Then g is **differentiable** at $p \in V$ if for some parametrization $f : U \subset \mathbb{R}^2 \to \Sigma$ with $p \in f(U) \subset V$, the composition $g \circ f : U \subset \mathbb{R}^2 \to \mathbb{R}$ is differentiable at $f^{-1}(p)$. g is **differentiable** in V if it is differentiable at all points of V.*

Definition 10 *Fix a parametrization $f : U \subset \mathbb{R}^2 \to \Sigma$ at $p \in \Sigma$, a regular surface. Then the **unit normal** at $q \in f(U)$ is defined as*

$$\mathbf{N}(q) = \frac{f_u \wedge f_v}{\|f_u \wedge f_v\|}(q)$$

Where f_u, f_v are the partial derivatives of f with respect to u and v respectively, and $\wedge$ is the usual vector product.

Definition 11 *A **tangent vector** to Σ at a point $p \in \Sigma$ is defined as the vector $\alpha'(0)$ of a differentiable parametrized curve $\alpha : (-\varepsilon, \varepsilon) \to \Sigma$ with $\alpha(0) = p$.*

Lemma 12 *Let $f : U \subset \mathbb{R}^2 \to \Sigma$ be a parametrization of a regular surface Σ and let $q \in U$. The vector subspace*

$$df_q(\mathbb{R}^2) \subset \mathbb{R}^3$$

coincides with the set of tangent vectors to Σ *at* $f(q) = p$. *Denote this tangent space by* $\mathbf{T}_p(\Sigma)$

Definition 13 *The quadratic form at* $p \in \Sigma$, *called the* ***First Fundamental Form*** *of* Σ *at* p, *denoted by* $\mathbf{I}_p$, *is defined as*

$$\mathbf{I}_p : \mathbf{T}_p(\Sigma) \to \mathbb{R} \text{ defined by}$$

$$\mathbf{I}_p(\mathbf{w}) = \langle \mathbf{w}, \mathbf{w} \rangle_p = |\mathbf{w}|^2$$

where $\mathbf{w} \in \mathbf{T}_p(\Sigma)$ *and* $\langle \mathbf{w}, \mathbf{w} \rangle_p$ *is the usual inner product of* $\mathbf{w}$ *with itself.*

Proposition 14 *If the tangent vector* $\mathbf{w}$ *coincides with the tangent vector to a parametrized curve* $\alpha(t) : (-\varepsilon, \varepsilon) \to \Sigma$ *where* $p = \alpha(0) = f(u_0, v_0)$ *then*

$$\mathbf{I}_p(\mathbf{w}) = E(u')^2 + Fu'v' + G(v')^2 \text{ where for } t = 0$$

$$E(u_0, v_0) = \langle f_u, f_u \rangle_p$$

$$F(u_0, v_0) = \langle f_u, f_v \rangle_p$$

$$G(u_0, v_0) = \langle f_v, f_v \rangle_p$$

Definition 15 *A regular surface* Σ *is* ***orientable*** *if it is possible to cover it with a family of coordinate neighborhoods in such a way that if a point* $p \in \Sigma$ *belongs to two neighborhoods of this family, the direction of the normal vector* $\mathbf{N}_p$ *with respect to these coordinates remains the same.* $\mathbf{N}$ *is the orientation of* Σ.

Definition 16 *Let* $\Sigma \subset \mathbb{R}^3$ *be a surface with orientation* $\mathbf{N}$. *The map* $\mathbf{N} : \Sigma \to \mathbb{R}^3$ *takes its values in the unit sphere*

$$\mathbf{S}^2 = \{(x, y, z) \in \mathbb{R}^3; |x^2 + y^2 + z^2 = 1\}$$

The map $\mathbf{N} : \Sigma \to \mathbf{S}^2$ *is called the* ***Gauss Map*** *of* Σ.

Definition 17 *The quadratic form* $\mathbf{II}_p(\mathbf{v})$ *defined on* $\mathbf{T}_p(\Sigma)$ *by* $\mathbf{II}_p(\mathbf{v}) = -\langle d\mathbf{N}_p, \mathbf{v} \rangle$ *is called the* ***Second Fundamental Form*** *of* Σ *at* p *and* $d\mathbf{N}_p$ *is the differential of the Gauss Map at* p.

Proposition 18 *Let* $\alpha(t) = f(u(t), v(t))$ *be a parametrized curve on* Σ *with* $\alpha(0) = p$. *The tangent*

vector to $\alpha(t)$ *at* p *is* $\alpha' = f_u u' + f_v v'$. *Then*

$$\mathrm{II}_p(\alpha') = e(u')^2 + gu'v' + h(v')^2 \text{ where}$$

$$e = -\langle \mathbf{N}_u, f_u \rangle = \langle \mathbf{N}, f_{\mathrm{u}\, u} \rangle$$

$$g = -\langle \mathbf{N}_v, f_u \rangle = \langle \mathbf{N}, f_{uv} \rangle = \langle \mathbf{N}, f_{vu} \rangle = -\langle \mathbf{N}_u, f_v \rangle$$

$$h = -\langle \mathbf{N}_v, f_v \rangle = \langle \mathbf{N}, f_{vv} \rangle$$

Proposition 19 *The value of* $\mathrm{II}_p(\mathbf{v})$ *for* $\mathbf{v} \in \mathrm{T}_p(\Sigma)$ *is equal to the normal curvature of a regular curve passing through* p *tangent to* $\mathbf{v}$. *In fact, all curves* α, *lying on the surface* Σ *such that for* $p \in \Sigma$, $\alpha(0) = p$ *and* $\alpha'(0) = \mathbf{v}$ *have the same normal curvature. We will denote the normal curvature by* k_n. *If we want to specify the normal curvature in a particular direction* $\mathbf{v}$, *we will denote it by* $k_n(\mathbf{v})$.

The the normal curvature is the normal component of the acceleration vector. It captures information about how the surface Σ forces a curve $\alpha(t)$ on the surface through p in a particular direction away from the tangent plane while disregarding other information about the curve.

Definition 20 *Let* $\{\mathbf{e}_1, \mathbf{e}_2\}$ *be the orthonormal basis of* $\mathrm{T}_p(\Sigma)$ *such that* $d\mathbf{N}_p(\mathbf{e}_1) = -k_1\mathbf{e}_1$ *and* $d\mathbf{N}_p(\mathbf{e}_2) = -k_2\mathbf{e}_2$. *Then* k_1, k_2 *are the maximum and minimum values of the Second Fundamental Form respectively restricted to the unit circle of* $\mathrm{T}_p(\Sigma)$. *They are, therefore, the extreme values of the normal curvature at* p, *called* ***Principal Curvatures***. *They are denoted by* $k_{\min_p}$ *and* $k_{\max_p}$. *Finally,* $\mathbf{e}_1$ *and* $\mathbf{e}_2$ *are called the* ***principal directions*** *at* p.

Proposition 21 *Let* $\mathbf{v} \in \mathrm{T}_p(\Sigma)$. *Then the normal curvature* k_n *along* $\mathbf{v}$ *is*

$$\mathrm{II}_p(\mathbf{v}) = k_n = k_1 \cos^2\theta + k_2 \sin^2\theta$$

where θ *is the angle between* $\mathbf{v}$ *and the principal direction* $\mathbf{e}_1$ *corresponding to* k_1.

Definition 22 *The* ***Gaussian Curvature*** K *is defined as*

$$K = k_1 k_2 = \frac{eh - g^2}{EG - F^2} = \det d\mathbf{N}_p$$

The mean curvature H *is defined as*

$$H = \frac{1}{2}(k_1 + k_2) = -\frac{1}{2} tr(d\mathbf{N}_p)$$

Definition 23 *A point of a surface Σ is*

1. *Elliptic if $K > 0$*
2. *Hyperbolic if $K < 0$*
3. *Parabolic if $K = 0$, with $d\mathbf{N}_p \neq 0$*
4. *Planar if $d\mathbf{N}_p = 0$*

These are some of the main concepts of Differential Geometry used in many of the reconstruction techniques we will discuss and explore. As we can see, each require that the surface be differentiable, which implies continuity. Recall, however, continuity does not imply differentiability. A surface can be continuous without being differentiable; the tip of a cone is one such example. Throughout the documentation, engineers use continuity and differentiability interchangeably, so care must be taken when reading the literature so we can correctly determine which they mean. Nevertheless, to directly utilize Differential Geometry, we must have a differentiable surface. The discrete set of points representing our surface is not differentiable, and before the surface is created, it is not even continuous. Therefore, we have to find ways to overcome the shortcomings of the point cloud. Basically, we can use the discrete data directly, or we can attempt to approximate the point cloud by a set of differentiable functions which will represent well known shapes with well-known normal and curvature values.

Engineers have come up with many methods by which to approximate the surface represented by the point cloud. In the following chapters, we will explore these methods. We will discuss various ways by which the normal vector is estimated, and the ways the normal vectors are used to recreate the surface. We will also discuss various methods by which curvature values are estimated.

3 NORMAL VECTORS

Perhaps the most important component of surface recreation is the establishment of a unit normal vector at each point on the point cloud PC or on our reconstructed surface S. Normal vectors are the most important component because they determine how the surface is oriented. Without well-estimated normal vectors, a non-oriented surface may be reproduced which bears little resemblance to the original surface. Because we are working with discrete data, and because most surfaces we want to reproduce will not have an associated parametrization, we cannot compute derivatives. Therefore, we are unable to utilize the definition from differential geometry of the normal vector directly. As we discussed earlier, there are two ways to overcome this shortcoming. We can either find a continuous function approximating points in a neighborhood of $p \in S$, or we can use the data directly by using statistical analysis, difference equations, and vector products.

3.1 Smooth Function Approximation

As we have discussed, we will consider methods by which normal vectors are found after the data has been manipulated. We will estimate the data by polynomials which describe well-known volumetric primitives, defined below, about a neighborhood of p. The primitives are then pieced together to form the computer generated surface. We can calculate the normal vector from these polynomials.

Definition 24 *A volumetric primitive is defined as a three dimensional shape having some volume associated with it. Some well-known volumetric primitives are spheres, cylinders and ellipsoids.*

Polynomial estimation methods are appealing in the fact that we can apply differential geometry directly to the polynomials. The drawback comes from the fact that only simple surfaces can be described by volumetric primitives. More complicated surfaces cannot be modeled by this method if the result is to be realistic. In some applications where realism is not essential it is preferred because it requires less computer time to recreate the surface. We will look at two methods developed by Ferrie [FR98] [FR93].

3.1.1 Method 1

Ferrie [FR93] approximates neighborhoods of $p \in PC$ by a continuous function and calculates the normal vector $\mathbf{N}_p$ from this function using the definition from Differential Geometry. For each $p \in PC$, we parameterize the points in the neighborhood of p by the quadratic

$$w(u,v) = au^2 + buv + cv^2$$

This function is found by a least squares technique determined by the user. This local parametrization is chosen so that the p is origin, i.e. $p = (0,0,0)$, of the parametric frame (u,v,w) with w aligning with the normal. Thus, we need an initial estimation of the normal. To accomplish this, we find a plane that best fits the data by any planar regression technique which is not discussed. Because this normal vector is simply the initial estimation, we only concern ourselves with the final estimation of $\mathbf{N}_p$.

Proposition 25 *Let the local parametrization of S at p be $f(u,v) = (u,v,w(u,v))$. Then*

$$\mathbf{N}_p = \frac{(-w_u, -w_v, 1)}{\sqrt{w_u^2 + w_v^2 + 1}}$$

Proof. This is straight from the definition of the normal vector from differential geometry and the proof of proposition 14. $\mathbf{N}_p = \frac{f_u \times f_v}{\|f_u \times f_v\|}$. So we have

$$f_u = (1, 0, w_u)$$

$$f_v = (0, 1, w_v)$$

$$\mathbf{N}_p = \frac{1}{\|f_u \times f_v\|} \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 1 & 0 & w_u \\ 0 & 1 & w_v \end{bmatrix}$$

$$= \frac{1}{\|f_u \times f_v\|} (\mathbf{e}_1(-w_u) - \mathbf{e}_2(w_v) + \mathbf{e}_3(1))$$

$$= \frac{(-w_u, -w_v, 1)}{\sqrt{w_u^2 + w_v^2 + 1}}$$

■

This completes our discussion of the first of two methods developed by Ferrie.

3.1.2 Method 2

The second method, [FR98], develops what he calls a "coarse" representation of the objects for the purposes of allowing robots to determine whether an object is in their path or allowing them to determine a method by which to grasp it. We begin with S represented as a set of discrete points with the coordinates in the form $(x, y, f(x, y))$ which is done by either the camera or a software program during preprocessing. We will approximate S with a function $\overline{f}$ based on the sample points in a local neighborhood of p. Using a local coordinate system with the origin (u, v, w) at $p \in S$ helps simplify the calculation. We want to define $\overline{f}$ so that it can fulfill the requirements of differential geometry, so we choose the function

$$\overline{f} = a_1u^3 + a_2u^2 + a_3u + a_4u^2v^2 + a_5u^2v + a_6uv^2 + a_7uv + a_8v^3 + a_9v^2 + a_{10}v + a_{11}$$

As with the method previously discussed, $\overline{f}$ is estimated using any least squares technique.

$\overline{f}$ allows us to use the definition of $\mathbf{N}_p$ from differential geometry by taking the derivatives and the vector product. Although many primitives can be created by this funciton, Ferrie addresses only spheres; we will do the same.

Proposition 26 *For a sphere,* $\mathbf{N}_p = \frac{(-a_3, -a_{10}, 1)}{\sqrt{a_3^2 + a_{10}^2 + 1}}$

Proof. The parametrization of a sphere is,

$$w(u, v) = (u, v, a_2u^2 + a_3u + a_9v^2 + a_{10}v + a_{11})$$

It is easy to see that this is a sphere by simply completing the square. Using the definition, we let $w_u(u, v)$, $w_v(u, v)$ be the derivative of w with respect to u, and v respectively. Thus

$$w_u(u, v) = (1, 0, 2a_2u + a_3)$$

$$w_v(u, v) = (0, 1, 2a_9v + a_{10})$$

We center the coordinate system so that $w(0, 0) = p$. By construction, p lies on the sphere so we

evaluate the vector product at $u = v = 0$

$$\begin{aligned}
\mathbf{N}_p &= \frac{w_u \times w_v}{\|w_u \times w_v\|} \\
&= \frac{1}{\|w_u \times w_v\|} \begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 1 & 0 & a_3 \\ 0 & 1 & a_{10} \end{vmatrix} \\
&= \frac{\mathbf{e}_1(-a_3) - \mathbf{e}_2(a_{10}) + \mathbf{e}_3}{\|\mathbf{e}_1(-a_3) - \mathbf{e}_2(a_{10}) + \mathbf{e}_3\|} \\
&= \frac{(-a_3, -a_{10}, 1)}{\sqrt{1 + a_3^2 + a_{10}^2}}
\end{aligned}$$

■

Unfortunately, Ferrie leaves the discussion here and does not pursue other volumetric primitives. Therefore, we will not be able to explore the methods he uses for other parametrizations. Although we suspect he uses the same approach. Without any further guidance, we cannot explore other volumetric primitives.

3.2 Discrete Data

Many methods have been developed that consider only the coordinates of the points in the PC. Each method considers a single point $p \in PC$ along with the nearest neighbors of p.

Definition 27 *Let $p \in PC$. We call the nearest k points to p, $k = 1, \ldots, m$, the **k-neighborhood** of p. The parameter k is usually set by the user, and we will denote this neighborhood as $nbhd(p)$.*

We might ask ourselves a few questions about this particular definition.

1. Why do we have the user specify the number of elements in the neighborhood rather than the metric size of the neighborhood? We specify the number of elements because we have more information at our disposal about the number of elements in the point cloud rather than how far apart they are from each other. It also gives the user more control over the speed of the reconstruction. Finally, it helps ensure that we have enough elements in the neighborhood to work with.

2. If there are two points, p_i, p_j, that are the same distance from p, how do we determine which point to consider an element of $nbhd\,(p)$? Suppose $k = m$, if we have not yet collected $(m-1)$ elements, both p_i and p_j will be considered in the neighborhood of p. (i.e. $p_i \in nbhd\,(p)$ and $p_j \in nbhd\,(p)$). However, if we have collected $(m-1)$ elements, and we only have room for one more element in $nbhd\,(p)$, we must decide which one to include. This process depends on the algorithm. If during the test $p_i \in nbhd\,(p)$ we have two possible situations, one of which would be included in the algorithm

$$\textbf{If } d\,(p,p_j) \leq d\,(p,p_i) \textbf{ Then } p_j \in nbhd\,(p) \text{ or}$$

In this case, if $d\,(p,p_j) = d\,(p,p_i)$ and we only have room for one more element in $nbhd\,(p)$ p_j will replace p_i. Thus, $p_j \in nbhd\,(p)$ but $p_i \notin nbhd\,(p)$. The other situation we have is

$$\textbf{If } d\,(p,p_j) < d\,(p,p_i) \textbf{ Then } p_j \in nbhd\,(p)$$

In this case, if $d\,(p,p_j) = d\,(p,p_i)$, p_j will not be included in $nbhd\,(p)$ if . Any algorithm that chooses elements based on distance, one of these two tests is necessary.

With these two questions answered, we can continue with our discussion about normal vectors.

3.2.1 Method 1

Hoppe et.al. [HOP92] finds the normal vector at a point $p \in PC$ through a *covariance matrix,* to be defined later. First, we will define the centroid of $nbhd(p)$.

Definition 28 *We call the* ***centroid of*** $nbhd(p)$ *the point*

$$o = \frac{1}{k}\sum_{i=1}^{k} p_i$$

where k is the number of elements in $nbhd(p)$.

Thus, the centroid o, is simply the average location of all the points in $nbhd(p)$, and is consistent with the mathematical definition of the centroid with weights equal to 1. The covariance includes the variance, which is defined in [BM95] as the average of the square of the distance that each values is from the expected location. Therefore, the values are real. In this case, we consider the

expected location to be the mean of the values. Thus, if the values are near the mean, the variance is small.

The following is the mathematical definition used for variance [GB91]

Definition 29 *The **variance** is defined as follows*

$$var = \frac{\sum(X-\mu)^2}{k}$$

where X is an individual value, μ is the mean of these values, and k is the population size

The covariance is the measurement of the mutual dependence of the variables (in statistical analysis, it measures the mutual dependence of the errors in those values). Now we will begin developing the covariant matrix. In this method, the covariant matrix is defined as follows:

Definition 30 *Let $p_i \in nbhd(p)$, $i = 1, \ldots, k$. The **Covariant Matrix** is defined as*

$$\mathbf{CM} = \sum_{i=1}^{k} (p_i - o) \otimes (p_i - o)$$

where if $(p_i - o) = (a, b, c)$, then

$$(p_i - o) \otimes (p_i - o) = \begin{bmatrix} aa & ab & ac \\ ba & bb & bc \\ ca & cb & cc \end{bmatrix}$$

Notice that the covariant matrix is simply the sum of squares of the difference between each p_i and the centroid o. For the purposes of 3-dimensional reconstruction, the properties of this matrix are well-suited for the information we can gain from it.

Proposition 31 **CM** *is symmetric, real, and positive semidefinite*

Proof. First, consider each $p_i \in nbhd(p)$, $i = 1, \ldots, k$, and let o be centroid of $nbhd(p)$. Since p_i is real for every i, o is also real. Therefore, $(p_i - o)$ is real, and $(p_i - o) \otimes (p_i - o)$ is real. Let $\mathbf{A}_i = (p_i - o) \otimes (p_i - o)$.

Before we proceed, we write $(p_i - o)$ as a real 3×1 matrix. Therefore, by definition, $(p_i - o)^T$ is a 1×3 matrix. We also know

$$(p_i - o) \otimes (p_i - o) = (p_i - o)\,(p_i - o)^T$$

and it is symmentric. To see this, suppose $(\mathbf{p}_i - \mathbf{o})^T = (a, b, c)$. Then

$$(p_i - o)(p_i - o)^T = \begin{pmatrix} a \\ b \\ c \end{pmatrix} (a, b, c)$$

$$= \begin{bmatrix} aa & ab & ac \\ ba & bb & bc \\ ca & cb & cc \end{bmatrix}$$

$$= (p_i - o) \otimes (p_i - o)$$

Now we want to show that $\mathbf{A}_i$ is positive semidefinite. Recall that a form $q(\mathbf{x})$ is positive semidefinite if $q(\mathbf{x}) \geq 0$ for all $\mathbf{x}$. In terms of matrices, this means that $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$ for all $\mathbf{x}$. So, our goal is to show that $\mathbf{x}^T (p_i - o)(p_i - o)^T \mathbf{x} \geq 0$ for all $\mathbf{x}$.

By [GB91], we see

$$\mathbf{x}^T (p_i - o)(p_i - o)^T \mathbf{x} = \left\langle (p_i - o)^T \mathbf{x}, (p_i - o)^T \mathbf{x} \right\rangle = \left\| (p_i - o)^T \mathbf{x} \right\|^2 \geq 0$$

This shows that $(p_i - o)(p_i - o)^T$ is semidefinite. Now, $(p_i - o)(p_i - o)^T$ is 3×3, and

$$rank\left((p_i - o)(p_i - o)^T\right) \leq rank(p_i - o) = 1 < 3$$

Thus, $(p_i - o)(p_i - o)^T$ is singular.

We will now show that the sum of real, symmetric, and positive semidefinite matrices results in a real, symmetric, positive semidefinite matrix. Since each $\mathbf{A}_i$ is real and symmetric,

$$\sum_{i=1}^{k} \mathbf{A}_i = \sum_{i=1}^{k} (p_i - o) \otimes (p_i - o) = \sum_{i=1}^{k} (p_i - o)(p_i - o)^T$$

is a real matrix because the sum of real numbers is real. Notice also that $\sum_{i=1}^{k} \mathbf{A}_i$ is symmetric, because

$$(\mathbf{A}_i + \mathbf{A}_j)^T = \mathbf{A}_i^T + \mathbf{A}_j^T = \mathbf{A}_i + \mathbf{A}_j$$

Finally, we need to show that the sum of positive semidefinite matrices results in a positive semidefinite matrix. By [ST86], we see that this is true. Again, consider $\mathbf{A}_i + \mathbf{A}_j$. Then

$$\mathbf{x}^T(\mathbf{A}_i + \mathbf{A}_j)\mathbf{x} = \mathbf{x}^T \mathbf{A}_i \mathbf{x} + \mathbf{x}^T \mathbf{A}_j \mathbf{x}, \text{ since}$$

$$\mathbf{x}^T\mathbf{A}_i\mathbf{x} \geq 0 \text{ and } \mathbf{x}^T\mathbf{A}_j\mathbf{x} \geq 0$$

$$\mathbf{x}^T\mathbf{A}_i\mathbf{x} + \mathbf{x}^T\mathbf{A}_j\mathbf{x} \geq 0$$

Thus, **CM** is real, symmetric, and positive semidefinite. ■

We now turn back to the method of finding the normal vectors in [HOP92]. After creating the covariance matrix, we decompose **CM** into its system of ordered eigenvalues which are greater than or equal to zero since the covariance matrix is positive semidefinite,

$$0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$$

The corresponding eigenvectors are

$$\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$$

where $\mathbf{e}_1$ is the eigenvector corresponding to λ_1, $\mathbf{e}_2$ is the eigenvector corresponding to λ_2, and $\mathbf{e}_3$ is the eigenvector corresponding to λ_3. We define the normal vector at p to be $\mathbf{N}_p = \pm\mathbf{e}_1$. The sign determines the orientation of the tangent plane, and must be chosen so nearby tangent planes are consistently oriented. In this method, the tangent planes are calculated next, so we will suspend this discussion to a later chapter.

3.2.2 Method 2

Gopi et.al. [GP00] utilizes the dot product and variance to approximate the normal vectors. Again, we utilize *nbhd*(p) defined earlier. For each $p_i \in nbhd(p)$ we define $\mathbf{v}_i$ be the vector from p to p_i.

$$\mathbf{v}_i = p_i - p$$

The normal vector to a surface at a point p and a vector at p tangent to the surface has a dot product equal to zero since they are orthogonal. According to Gopi et.al., our approximated normal vector, $\mathbf{N}_p$, will be the vector up to a sign that minimizes the cosine of the angle from each $\mathbf{v}_i$ to $\mathbf{N}_p$. We will substitute the dot product in the definition of variance to obtain the definition by Gopi [GP00]

Definition 32 *The Variance is defined as follows*

$$var\,(p) = \sum_{i=1}^{k} \frac{\left(\mathbf{N}_p \cdot \mathbf{v}_i - \frac{\sum_{i=1}^{k} \mathbf{N}_p \cdot \mathbf{v}_i}{k}\right)^2}{k}$$

Minimizing the dot product will minimize the variance. Removing the factor $\frac{1}{k}$ will not change which vector minimizes the variance, so we can remove the factor $\frac{1}{k}$ and minimize to obtain

$$\begin{aligned}\min\left(var\left(p\right)\right) &= \min\left(\sum_{i=1}^{k}\left(\mathbf{N}_p\cdot\mathbf{v}_i - \frac{\sum_{i=1}^{k}\mathbf{N}_p\cdot\mathbf{v}_i}{k}\right)^2\right)\\ &= \min\left(\sum_{i=1}^{k}\left(\left(\mathbf{v}_i - \frac{\sum_{i=1}^{k}\mathbf{v}_i}{k}\right)\cdot\mathbf{N}_p\right)^2\right)\end{aligned}$$

Proposition 33 *Arrange the coordinates of the nbhd(p) so that p is the origin. Then*

$$\min\left(var\left(p\right)\right) = \min\left(\sum_{i=1}^{k}\left(\left(\mathbf{v}_i-\mathbf{c}\right)\cdot\mathbf{N}_p\right)^2\right)$$

where $\mathbf{c}$ *is the centroid of nbhd(p).*

Proof. Since p is the origin, the centroid, with all weights equal to 1, we have

$$\mathbf{c} = \frac{\sum_{i=1}^{k}p_i}{k}$$

Since $\mathbf{v}_i = p_i - p$, we have

$$\mathbf{c} = \frac{\sum_{i=1}^{k}\mathbf{v}_i}{k}$$

Thus

$$\begin{aligned}\min(var\left(p\right)) &= \min\left(\sum_{i=1}^{k}\left(\left(\mathbf{v}_i - \frac{\sum_{i=1}^{k}\mathbf{v}_i}{k}\right)\cdot\mathbf{N}_p\right)^2\right)\\ &= \min\left(\sum_{i=1}^{k}\left(\left(\mathbf{v}_i-\mathbf{c}\right)\cdot\mathbf{N}_p\right)^2\right)\end{aligned}$$

■

Define $\mathbf{A}$ to be the $k\times 3$ matrix defined by row vectors $\mathbf{v}_i - \mathbf{c}$. We thus obtain

$$\begin{aligned}\min(var\left(p\right)) &= \min\left(\left\|\mathbf{A}\mathbf{N}_p\right\|_2\right)\\ &= \min\left(\mathbf{N}_p^T\mathbf{A}^T\mathbf{A}\mathbf{N}_p\right)^{\frac{1}{2}}\end{aligned}$$

It is claimed that the vector which minimizes the variance is the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}$. The difficulty with this assertion is that $\mathbf{A}$ is not square. By definition, we can take the square root of the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$. Another assertion is that it is the smallest singular value in the singular value decomposition [GB91][GL83].

Lemma 34 *The vector that minimizes* $\|\mathbf{AN}_p\|_2$ *is the eigenvector corresponding to the smallest eigenvalue of* $\mathbf{A}^T\mathbf{A}$ *up to a sign. It is also the vector, up to sign, corresponding smallest value in the SVD of* $\mathbf{A}$.

Proof. The first part of this lemma is clear because by definition $\|\mathbf{AN}_p\|_2 = \left(\mathbf{N}_p^T\mathbf{A}^T\mathbf{AN}_p\right)^{\frac{1}{2}}$, so naturally, the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$ minimizes $\|\mathbf{AN}_p\|_2$. The second part is straight forward from the SVD Theorem and proof immediately following. ■

To explore the second part of the lema, we will recall the Singular Value Decomposition Theorem. We will denote real $m \times n$ matrices by $\mathbb{R}^{m\times n}$, and adapt the proof from [GB91].

Theorem 35 *Singular Value Decomposition (SVD). If* $\mathbf{A} \in \mathbb{R}^{k\times 3}$ *then there exist orthogonal matrices*

$$\mathbf{U} = [u_1, u_2, \ldots, u_m] \in \mathbb{R}^{k\times k} \text{ and}$$
$$\mathbf{V} = [v_1, v_2, \ldots, v_n] \in \mathbb{R}^{3\times 3} \text{ such that}$$
$$\mathbf{U}^T\mathbf{AV} = diag(\lambda_1, \lambda_2, \lambda_3, \mathbf{0}) = \mathbf{S} \text{ where}$$
$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$$

Since $\mathbf{U}$ *and* $\mathbf{V}$ *are orthogonal and real, we can write*

$$\mathbf{A} = \mathbf{USV}^H = \mathbf{USV}^T$$

where the columns of $\mathbf{U}$ *are the eigenvectors of* $\mathbf{AA}^T$ *and the columns of* $\mathbf{V}$ *are the eigenvectors of* $\mathbf{A}^T\mathbf{A}$

Proof. Since $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$ are Hermitian by definition, symmetric, i.e. $\left(\mathbf{AA}^T\right)^T = \mathbf{A}^{T^T}\mathbf{A}^T = \mathbf{AA}^T$, and positive semidefinite matrices, the nonzero eigenvalues are positive. We define the positive square roots of these eigenvalues as the singular values of $\mathbf{A}$. Let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ be the eigenvectors of $\mathbf{A}^T\mathbf{A}$ and let $\mathbf{V}$ be the matrix with the columns as the eigenvectors of $\mathbf{A}^T\mathbf{A}$

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}$$

So by construction, $\mathbf{V}$ is a 3×3 matrix whose columns are orthonormal vectors that form an orthonormal basis for $\mathbb{R}^3$. Suppose $rank(\mathbf{A}) = r$. Then WLOG we can assume the first r columns

of $\mathbf{V}$ are eigenvectors associated with the nonzero eigenvalues of $\mathbf{A}^T\mathbf{A}$. In other words,

$$\mathbf{A}^T\mathbf{A}\mathbf{v}_i = \lambda_i^2\mathbf{v}_i, 1 \leq i \leq r \leq 3$$

The remaining $3-r$ columns in $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$ corresponding to its zero eigenvalue.

We will now construct $\mathbf{U}$. For $1 \leq i \leq r$, set

$$\mathbf{u}_i = \left(\frac{1}{\lambda_i}\right)\mathbf{A}\mathbf{v}_i$$

Consider any two vectors $\mathbf{u}_i$ and $\mathbf{u}_j$. Then

$$\begin{aligned}\langle \mathbf{u}_i, \mathbf{u}_j \rangle &= \left(\frac{1}{\lambda_i}\right)\left(\frac{1}{\lambda_j}\right)\langle \mathbf{A}\mathbf{v}_i, \mathbf{A}\mathbf{v}_j \rangle = \left(\frac{1}{\lambda_i}\right)\left(\frac{1}{\lambda_j}\right)\langle \mathbf{v}_i, \mathbf{A}^T\mathbf{A}\mathbf{v}_j \rangle \\ &= \left(\frac{1}{\lambda_i}\right)\left(\frac{1}{\lambda_j}\right)\langle \mathbf{v}_i, \lambda_j^2\mathbf{v}_j \rangle = \left(\frac{\lambda_j}{\lambda_i}\right)\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \left(\frac{\lambda_j}{\lambda_i}\right)\delta_{ij}\end{aligned}$$

since $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 1$ if $i = j$, and $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$ if $i \neq j$.

So the set $\{\mathbf{u}_1, \ldots, \mathbf{u}_r\}$ is orthonormal. The orthonormal vectors compose the first r columns of $\mathbf{U}$. The remaining $k-r$ columns of $\mathbf{U}$ are orthonormal vectors forming a basis for $null(\mathbf{A}\mathbf{A}^T)$, the eigenspace of $\mathbf{A}\mathbf{A}^T$ corresponding to $\lambda = 0$.

Finally, we calculate $\mathbf{U}^T\mathbf{A}\mathbf{V}$

$$\begin{aligned}\mathbf{U}^T\mathbf{A}\mathbf{V} &= \mathbf{U}^T\mathbf{A}\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \\ &= \mathbf{U}^T\begin{bmatrix} \mathbf{A}\mathbf{v}_1 & \mathbf{A}\mathbf{v}_2 & \mathbf{A}\mathbf{v}_3 \end{bmatrix} \\ &= \mathbf{U}^T\begin{bmatrix} \lambda_1\mathbf{u}_1 & \lambda_2\mathbf{u}_2 & \lambda_3\mathbf{u}_3 & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1\mathbf{U}^T\mathbf{u}_1 & \lambda_2\mathbf{U}^T\mathbf{u}_2 & \lambda_3\mathbf{U}^T\mathbf{u}_3 & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1\mathbf{e}_1 & \lambda_2\mathbf{e}_2 & \lambda_3\mathbf{e}_3 & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix} = \mathbf{S}\end{aligned}$$

where $\mathbf{e}_i$ is the $k \times 1$ vector with a 1 in the i^{th} row

So we see that $\mathbf{S} \in \mathbb{R}^{k \times 3}$ matrix with λ_i, $1 \leq i \leq r$ along the diagonal and zero everywhere else. Thus, the smallest eigenvalue is the smallest value in the SVD of $\mathbf{A}$, and the corresponding eigenvector is the vector minimizing $\|\mathbf{A}\mathbf{N}_p\|_2$. ■

SVD is a diagonalization algorithm for any matrix $\mathbf{A}$ which utilizes the eigenvalues $\mathbf{A}^T\mathbf{A}$ for its development. The purpose for specifying the fact that it is the smallest value in the SVD of $\mathbf{A}$ is unnecessary. Calculating the eigenvalues directly from $\mathbf{A}^T\mathbf{A}$ would be simpler and more efficient.

Lemma 36 *Methods 1 and 2 will result in the same normal vectors.*

Proof. The centroid in method 1 is defined as

$$o = \frac{1}{k}\sum_{i=1}^{k} p_i$$

where $p_i \in nbhd\,(p)$. In method 2, we define a vector

$$\mathbf{v}_i = p_i - p$$

Then the centroid is defined as

$$\mathbf{c} = \frac{1}{k}\sum_{i=1}^{k} p_i$$

Let p be the origin. Then $\mathbf{c} = o$.

The covariance matrix in method 1 is defined as

$$\mathbf{CM} = \sum_{i=1}^{k} (p_i - o) \otimes (p_i - o)$$

where if $(p_i - o) = (a_i, b_i, c_i)$, then

$$\sum_{i=1}^{k} (p_i - o) \otimes (p_i - o) = \sum_{i=1}^{k} \begin{bmatrix} a_i^2 & a_i b_i & a_i c_i \\ b_i a_i & b_i^2 & b_i c_i \\ c_i a_i & c_i b_i & c_i^2 \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{k} a_i^2 & \sum_{i=1}^{k} a_i b_i & \sum_{i=1}^{k} a_i c_i \\ \sum_{i=1}^{k} b_i a_i & \sum_{i=1}^{k} b_i^2 & \sum_{i=1}^{k} b_i c_i \\ \sum_{i=1}^{k} c_i a_i & \sum_{i=1}^{k} c_i b_i & \sum_{i=1}^{k} c_i^2 \end{bmatrix}$$

$$= \mathbf{A}^T\mathbf{A}$$

Thus, finding the eigenvector which minimizes $var\,(p) = \|\mathbf{A}\mathbf{e}_p\|$ will also minimize $\mathbf{CM}$. ∎

Corollary 37 $\mathbf{N}_p$ *minimizes the dot product of the vectors making up the rows of* $\mathbf{A}$.

Proof. Suppose $\|\mathbf{A}\mathbf{e}_p\|$ is minimized for $\mathbf{e}_p = \mathbf{N}_p$. Then

$$\mathbf{A}\mathbf{N}_p = \begin{bmatrix} \langle (p_1 - o), \mathbf{N}_p \rangle \\ \langle (p_2 - o), \mathbf{N}_p \rangle \\ \vdots \\ \langle (p_k - o), \mathbf{N}_p \rangle \end{bmatrix}$$

is minimum. ■

Therefore, in both cases, we have a good estimate of the normal vector.

3.2.3 Method 3

We now turn to another method used to estimate the normal vectors. Taubin [TB95] discusses primarily curvature. However, to calculate the curvature, we first approximate a normal vector. Before we begin, we will set up some notation.

This method begins with a triangulation. It does not establish a triangulation, it simply takes a triangulation T as input and calculates curvature from it. The triangulated surface S consists of a list of vertices P and faces F. Denote this list by

$$S = \{P, F\}$$

$$P = \{p_i : 1 \leq i \leq n\}$$

n is the number of vertices.

$$F = \{f_j : 1 \leq j \leq m\}$$

m is the number of faces.

Each face consists of three nonrepeating vertices, so

$$f_i = (p_{1_i}, p_{2_i}, p_{3_i})$$

$$p_{j_i} \neq p_{l_i} \text{ when}$$

$$j \neq l \text{ for}$$

$$1 \leq j \leq 3$$

$$1 \leq l \leq 3$$

We assume S is oriented. The set of vertices sharing a face with p_i will be denoted P^i. Thus if p_j belongs to P^i, p_j is a neighbor to p_i. The set of faces containing vertex p_i will be F^i. If f_k belongs to F^i, f_k is said to be incident to p_i.

Remember that we are beginning with a triangulation, T, of a surface. This means that the surface has already been approximated by a set of planar triangles. From basic calculus, there is a well defined normal vector up to orientation on each triangle, call it $\mathbf{N}_{f_i}$. Recall that for face f_i consisting of vertices $(p_{1_i}, p_{2_i}, p_{3_i})$, where $\mathbf{v}_1 = p_{2_i} - p_{1_i}$ and $\mathbf{v}_2 = p_{3_i} - p_{1_i}$, then the normal vector to the planar triangle f_i is $\mathbf{N}_{f_i} = \mathbf{v}_1 \times \mathbf{v}_2$.

Since we are assuming that S is oriented, every $\mathbf{N}_{f_i}$ points to the same side of the surface. However, we do not want the normal vector of each face of T, we want to estimate the normal vector at each vertex.

Definition 38 *Let $\mathbf{N}_{p_i}$, the approximated normal vector at point $p_i \in S$, be defined as follows:*

$$\mathbf{N}_{p_i} = \frac{\sum_{f_i \in F^i} |f_i| \mathbf{N}_{f_i}}{\left\| \sum_{f_i \in F^i} |f_i| \mathbf{N}_{f_i} \right\|}$$

where $|f_i|$ is the area of the face f_i.

So by definition, the normal vector at p_i is defined as the normalized weighted sum of the normal vectors of the incident faces. Notice that the larger the incident face, the more weight that normal has in the overall computation. In ordinary situations, this calculation would be sufficient because we would end up with a normal vector that would average out over all the incident faces.

3.2.4 Method 4

Triangulations are not the only discrete recreation methods which require normal vector calculations. Delingette [DEL94] utilizes a generalized polynomial mesh, called a simplex mesh, to recreate the surface. We categorize a simplex mesh according to the number of vertices each element in the point cloud is connected to. An N-Simplex Mesh connects each element in the point cloud to $(N+1)$ vertices. Thus, a 1-Simplex Mesh connects each element in the point cloud to two other elements, a 2-Simplex Mesh connects each element to three other elements. It is represented by a list of pairs consisting of each vertex and the function connecting it to its three nearest neighbors. The

connecting function is defined as a distance function, but is not discussed in detail. To define it more precisely,

Definition 39 *A 2* ***simplex mesh,*** $(2SM)$*, of* $\mathbb{R}^3$ *is defined as a pair* $(V(S), N(S))$ *where*

1. $V(S) = \{p_i\}$, $i = 1, \ldots n$, *is the set of all vertices making up* S.
2. $N(S) : p_i \rightarrow \{p_{i_1}, p_{i_2}, p_{i_3}\}$. $N(S)$ *is a function that connects each* p_i *is to three neighbors*
3. $\forall i \in \{1, \ldots n\}, \forall j \in \{1, 2, 3\}, \forall l \in \{1, 2, 3\}$

$$p_{i_j} \neq p_i$$

$$p_{i_j} \neq p_{i_k}$$

So p_i *is not connected to itself, and* p_i *is connected to three distinct neighbors.*

This method has many factors to it, most of which we will not explore because it utilizes virtual reality technology to recreate the surface. However, we do want to notice a few properties.

First, we observe that a 2-simplex mesh can be obtained from a triangulation by connecting the centers of each triangle in the triangulation, see figure 4 adapted from [DEL94]. Thus, it is a simple manner to estimate the normal vector on each face.

Definition 40 *Let* S *be an oriented simplex mesh representing* Σ. *Let* $p_i \in S$ *be a vertex of a simplex mesh and let* $(p_{i_1}, p_{i_2}, p_{i_3})$ *be the three neighbors of* p_i *on the mesh. These three neighbors define a plane* P *with a normal vector*

$$\mathbf{N}_P = \frac{(p_i - p_{i_1}) \times (p_i - p_{i_2}) + (p_{i_2} \times p_{i_3}) + (p_{i_1} \times p_{i_3})}{\|(p_i - p_{i_1}) \times (p_i - p_{i_2}) + (p_{i_2} \times p_{i_3}) + (p_{i_1} \times p_{i_3})\|}$$

We define the normal vector $\mathbf{N}_{p_i} = \mathbf{N}_P$.

Thus, the normal vector at point p_i is simply the normal of the plane defined by its neighbors in the $2SM$, see figure 5.

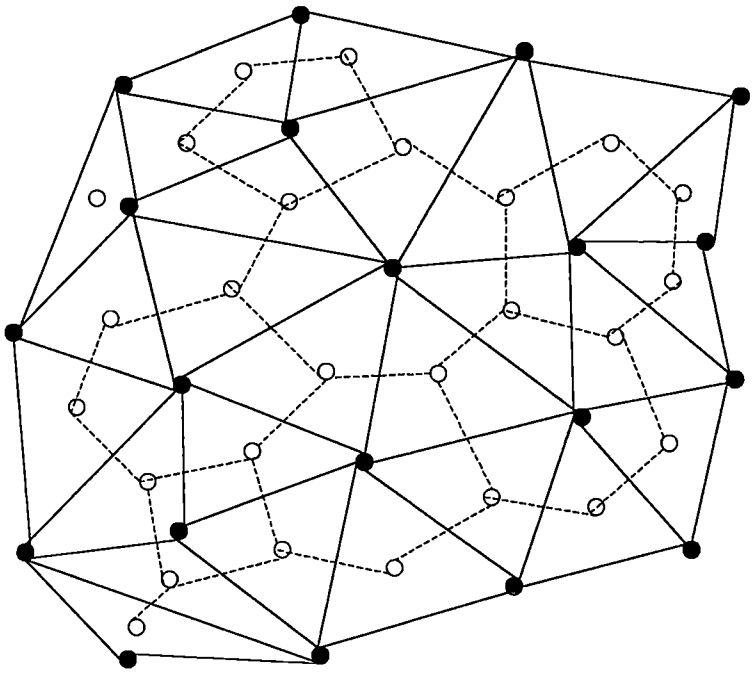

Figure 4: A Simplex Mesh from a Triangulation

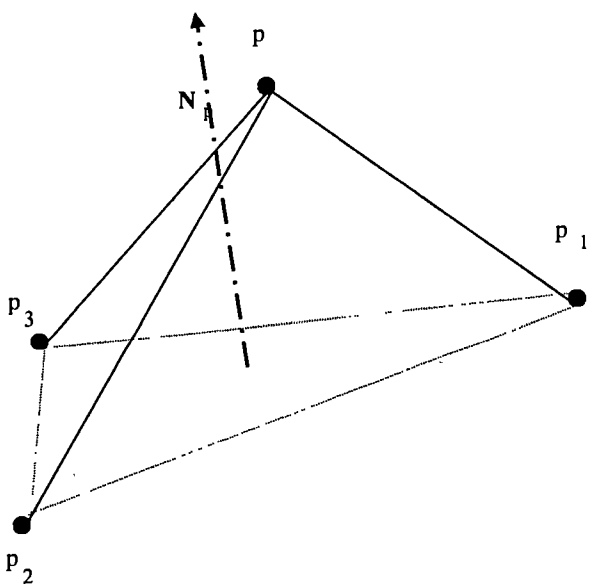

Figure 5. The Normal at p_i on a 2 Simplex Mesh

This definition of the normal of the plane determined by $\{p_{i_1}, p_{i_2}, p_{i_3}\}$ is a common algorithm, and can be found in [MIL77]. Notice that we could simply take a single cross product to obtain the same normal vector. It works well in this setting because the surfaces are not assumed to have radical changes in the normal vector about the neighborhood of p_i. We know this vector will always exist unless the plane is degenerate. This will only happen if the neighbors are colinear. Because the chances of this happening are very small given that the neighbors are determined via a distance function, this possibility is not addressed. Furthermore, we can ensure non-collinearity by choosing $N(S)$ carefully.

This completes our discussion of methods used to estimate normal vectors. In the next section, we will revisit these methods to determine which ones may be more accurate.

3.3 Conclusion

We discussed two general methods developed by engineers to estimate normal vectors. Although the details may be different, the overall approaches between the two types of methods are very similar. Which is more effective depends primarily on the application and the speed required to recreate the object. Although a thorough discussion of the reliability of these methods must wait until our final discussion, a preliminary discussion restricted to the normal vector calculations is possible.

The methods involving difference equations and statistical analysis generally produce very similar results, and as we have found, two of these methods produce the same Normal vector. Nevertheless, all of them measure how $p_i \in nbhd(p)$ differs from p in each coordinate, collect this information in a covariant matrix **CM**, and use this information to determine $\mathbf{N}_p$. All cases define $\mathbf{N}_p = \pm\mathbf{e}_1$ the eigenvector corresponding to the smallest eigenvalue of **CM**. They choose this vector because it is the one that will *minimize* **CM**, which minimizes the inner product.

We also looked at a couple of methods which approximate the data with continuous functions and use those functions to estimate the normal vectors. By utilizing continuous functions, we can calculate the normal vector by using the definition from differential geometry. Because this approach is simpler, one might believe these methods would be preferable to the discrete case described in the previous paragraph, but this is not the case. The discrete case will produce normal vectors that more closely resemble the normal vectors of the original surface Σ because we do not restrict

ourselves to specific smooth surfaces. The results produced by an approximating function will produce normal vectors based on the surface represented by the smooth function rather than the data itself. If the surface is complicated, these primitives may not be an accurate representation of Σ. As a result of these differences, normal vector approximation based on the discrete data directly is preferrable.

4 CURVATURE

The shape of a surface is closely linked to its curvature. If the shape changes dramatically, the curvature is highly positive or negative; if it changes only slightly, the curvature is near zero. For example, a plane does not change its shape at all, so it has curvature equal to zero. As we discussed in chapter 2, there are different measurements of curvature:

- Principal Curvatures
- Gaussian Curvatures
- Mean Curvature
- Normal Curvature

Although curvature is only used in a limited way during surface reconstruction, it is used extensively in other aspects of computer imaging. After a computer image has been generated, we often wish to partition the surface into its constituent parts to facilitate manipulation or recognition. This process is called surface segmentation and curvature estimation is used extensively in this process. A surface is segmented based on curvature based on the idea that when two surfaces come together, they meet at a deep concavity. Curvature values are also used in surface recognition methods.

Curvature can be estimated in different ways. One approach is a watershed segmentation technique developed by Mangan and Whitaker [MN99]. This approach simulates water filling crevices in the triangulation. The crevices which hold the most water are estimated to be the areas of lowest curvature. Another approach simulates electrical charge distribution over the triangulation. This approach developed by Wu and Levine [Wu97] is based on the idea that on a perfect conductor, the charge density in areas of low curvature is a local minima. We will not be discussing these approaches; instead, we will concern ourselves only with more traditional mathematical approaches to calculating curvature.

Two of these traditional methods are used to segment the point cloud itself into partitions. We locate these partitions so primitives can be fitted to each one [FR93][FR98]. In one of these methods, [FR93] the principal curvature values are refined according to neighbors. In another

method, we estimate curvature to set sampling criteria [GP00]. This chapter will be concerned with these methods as well as other methods that estimate curvature on a triangulation or simplex mesh [DEL94].

We will begin with methods that calculate curvature on smooth functions.

4.1 Smooth Function Approximation

Recall that in both [FR93][FR98] continuous functions were used to approximate the data in the point cloud by some least squares technique. We will begin with the more general approach [FR98]. The the local approximating function is of the form

$$w(u,v) = a_1u^3 + a_2u^2 + a_3u + a_4u^2v^2 + a_5u^2v + a_6uv^2 + a_7uv + a_8v^3 + a_9v^2 + a_{10}v + a_{11}$$

Thus, for each point $p \in PC$ we have

$$\overline{f}(u,v) = (u,v,w(u,v))$$

We will use $\overline{f}$ and the second fundamental form to calculate the principal curvatures

Lemma 41 *The principal curvatures $k_{\max_p}$ and $k_{\min_p}$ are*

$$k_{\max_p} = a_2 + a_9 + \sqrt{(a_2+a_9)^2 + a_7^2}$$
$$k_{\min_p} = a_2 + a_9 - \sqrt{(a_2+a_9)^2 + a_7^2}$$

The corresponding principal directions are

$$\mathbf{e}_{1_p} = \begin{bmatrix} 1 \\ -\frac{\sqrt{a_9^2 - a_2a_9 + a_7^2 + a_2^2} - a_9 + a_2}{a_7} \end{bmatrix}$$

$$\mathbf{e}_{2_p} = \begin{bmatrix} 1 \\ \frac{\sqrt{a_9^2 - a_2a_9 + a_7^2 + a_2^2} + a_9 - a_2}{a_7} \end{bmatrix}$$

Proof. First, we use the basic definition of the coefficients of the second fundamental form. DoCarmo [DC76] shows us that

$$\mathbf{N}_p(u,v) = \frac{(-w_u, -w_v, 1)}{\sqrt{1+w_u^2+w_v^2}}$$

$$e = \frac{w_{u\,u}}{\sqrt{1+w_u^2+w_v^2}}$$

$$g = \frac{w_{uv}}{\sqrt{1+w_u^2+w_v^2}}$$

$$h = \frac{w_{vv}}{\sqrt{1+w_u^2+w_v^2}}$$

Because the local coordinate system is centered at p, we see that $w_u(0,0) = 0$ and $w_v(0,0) = 0$.

By a simple calculation, we also observe that

$$\overline{f}_{u\,u} = (0,0,w_{u\,u}) = \left(0,0,6a_1u + 2a_2 + 2a_4v^2 + 2a_5v\right)$$

$$\overline{f}_{uv} = (0,0,w_{uv}) = (0,0,4a_4uv + 2a_5u + 2a_6v + a_7)$$

$$\overline{f}_{vv} = (0,0,w_{vv}) = \left(0,0,2a_4u^2 + 2a_6u + 6a_8v + 2a_9\right)$$

Now, evaluate $\overline{f}_{u\,u}, \overline{f}_{u\,v}, \overline{f}_{vv}$ with $u = v = 0$. Thus

$$e = \frac{w_{u\,u}}{\sqrt{1+w_u^2+w_v^2}} = 2a_2$$

$$g = \frac{w_{uv}}{\sqrt{1+w_u^2+w_v^2}} = a_7$$

$$h = \frac{w_{vv}}{\sqrt{1+w_u^2+w_v^2}} = 2a_9$$

Thus, the second fundamental form applied to vector $\mathbf{v} = (u,v)$ becomes

$$\mathbf{II}_p(\mathbf{v}) = edu^2 + 2gdudv + hdv^2$$

$$= \begin{bmatrix} du & dv \end{bmatrix} \begin{bmatrix} e & g \\ g & h \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix}$$

$$= \mathbf{v}^T \begin{bmatrix} 2a_2 & a_7 \\ a_7 & 2a_9 \end{bmatrix} \mathbf{v}$$

By definition, the eigenvalues of $\mathbf{A}$ are the principal curvatures

$$k_{\max_p} = a_2 + a_9 + \sqrt{(a_2 + a_9)^2 + a_7^2}$$
$$k_{\min_p} = a_2 + a_9 - \sqrt{(a_2 + a_9)^2 + a_7^2}$$

and the eigenvectors are the principal directions

$$\mathbf{e}_{1_p} = \begin{bmatrix} 1 \\ -\frac{\sqrt{a_9^2 - a_2 a_9 + a_7^2 + a_2^2} - a_9 + a_2}{a_7} \end{bmatrix}$$

$$\mathbf{e}_{2_p} = \begin{bmatrix} 1 \\ \frac{\sqrt{a_9^2 - a_2 a_9 + a_7^2 + a_2^2} + a_9 - a_2}{a_7} \end{bmatrix}$$

where $\mathbf{e}_{1_p}$ corresponds to $k_{\max_p}$ and $\mathbf{e}_{2_p}$ corresponds to $k_{\min_p}$. This completes the proof. ■

In the second method developed by Ferrie, [FR93] we began with quadratic functions with a local coordinate system defined with the origin at p. The quadratic had the form

$$w(u,v) = au^2 + buv + cv^2$$

Notice that this is simply a special case of the previous method. In the previous method, we began with the approximating function of the form

$$w(u,v) = a_1u^3 + a_2u^2 + a_3u + a_4u^2v^2 + a_5u^2v + a_6uv^2 + a_7uv + a_8v^3 + a_9v^2 + a_{10}v + a_{11}$$

If

$$a_1 = a_3 = a_4 = a_5 = a_6 = a_8 = a_{10} = a_{11} = 0$$

we obtain our current approximating function. We approximate the point cloud with

$$\overline{f}(u,v) = (u, v, w(u,v))$$

We can now calculate the principal curvatures and corresponding directions using the same approach.

Lemma 42 *Let $\overline{f}(u,v) = (u, v, w(u,v))$. Then the maximum and minimum principal curvatures, $k_{\min_p}$ and $k_{\max_p}$, are given by*

$$k_{\min_p} = a + c + \sqrt{(a-c)^2 + b^2}$$
$$k_{\max_p} = a + c - \sqrt{(a - c^2 + b^2)}$$

and the corresponding principal directions are

$$\mathbf{e}_{1_p} = \begin{bmatrix} b \\ -\left(a-c+\sqrt{(a-c)^2+b^2}\right) \end{bmatrix}$$

and

$$\mathbf{e}_{2_p} = \begin{bmatrix} -b \\ a-c+\sqrt{(a-c)^2+b^2} \end{bmatrix}$$

Proof. Notice that from the previous corollary

$$a = a_2$$

$$b = a_7$$

$$c = a_9$$

Thus

$$k_{\max_p} = a+c+\sqrt{(a-c)^2+b^2}$$
$$k_{\min_p} = a+c-\sqrt{(a-c)^2+b^2}$$

This concludes the curvature portion of our proof. Next, the principal directions in the previous lemma are

$$\mathbf{e}_{1_p} = \begin{bmatrix} 1 \\ -\frac{\sqrt{a_9^2-a_2a_9+a_7^2+a_2^2}-a_9+a_2}{a_7} \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{\sqrt{c^2-ac+b^2+a^2}-c+a}{b} \end{bmatrix}$$

$$\mathbf{e}_{2_p} = \begin{bmatrix} 1 \\ \frac{\sqrt{a_9^2-a_2a_9+a_7^2+a_2^2}+a_9-a_2}{a_7} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{\sqrt{c^2-ac+b^2+a^2}+c-a}{b} \end{bmatrix}$$

Multiplying both by $-b$ will not alter the direction of the vector. Therefore, we obtain

$$\mathbf{e}_{1_p} = \begin{bmatrix} b \\ -\left(a-c+\sqrt{c^2-ac+b^2+a^2}\right) \end{bmatrix}$$

$$\mathbf{e}_{2_p} = \begin{bmatrix} -b \\ a-c+\sqrt{c^2-ac+b^2+a^2} \end{bmatrix}$$

This completes our proof. ■

These methods for estimating the principal curvatures at a point p when given a smooth function approximation of our data simply use definitions and theorems from Differential Geometry. However, to obtain reliable estimates of $k_{\min_p}$ and $k_{\max_p}$, the choice of parameterizations is critical. It is for this reason, that the parametrizations are defined only in very small neighborhoods defined by the user. The specific method for defining the neighborhood is not discussed.

We will now continue with curvature determination from discrete data.

4.2 Curvature Estimation from Discrete Data

In the next two sections, we will discuss methods estimating curvature based on the mesh representing the surface. We will first discuss approximations of principal and normal curvatures, then we will discuss methods by which to estimate Gaussian and Mean curvatures.

4.2.1 Method 1

Two methods, Taubin [TB95] and Gopi et.al. [GP00], estimate the curvature tensor and use it to estimate the principal curvatures. We will approach this method in two distinct steps. First, we will discuss the curvature tensor and related curvature values in the continuous case. Second, we will use the continuous case as our foundation to develop a method to estimate these values from discrete data.

Curvature Tensor in the Continuous Case The curvature tensor is simply a map which assigns each point p on a surface Σ to a function that measures the normal curvature. Taubin [TB95] states the definition of normal curvature as follows:

Definition 43 *The **Normal Curvature**, $k_{n_p}(\mathbf{v})$ at a point $p \in \Sigma$ in the direction of a unit tangent vector $\mathbf{v}$ is defined by $x''(0) = k_{n_p}(\mathbf{v})\,\mathbf{N}_p$, where $x(s)$ is a curve on Σ at p parametrized by arc length such that $x(0) = p$ and $x'(0) = \mathbf{v}$*

Notice that this is not the same as the mathematical definition of normal curvature. The tangental component of the curve is ignored. By this definition the normal curvature is a quadratic

form which satisfies the identity

$$k_{n_p}(\mathbf{v}) = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} k_p^{11} & k_p^{12} \\ k_p^{21} & k_p^{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

where $\mathbf{v}$ is a tangent vector to Σ at p. We can represent $\mathbf{v}$ as a linear combination of an orthonormal basis $\{\mathbf{v}_1, \mathbf{v}_2\}$of the tangent space to Σ at p. Thus, $\mathbf{v} = v_1\mathbf{v}_1 + v_2\mathbf{v}_2$ and

$$\begin{aligned} k_p^{11} &= k_{n_p}(\mathbf{v}_1) \\ k_p^{22} &= k_{n_p}(\mathbf{v}_2) \\ k_{n_p}^{12} &= k_{n_p}^{21} \end{aligned}$$

We want our basis $\{\mathbf{v}_1, \mathbf{v}_2\}$ to consist of the principal directions $\{\mathbf{e}_1, \mathbf{e}_2\}$; thus $k_p^{12} = k_p^{21} = 0$, and we obtain

$$\begin{aligned} k_{n_p}(\mathbf{v}) &= k_{n_p}\left(\begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} \right) \\ &= \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} k_{n_p}(\mathbf{e}_1) & 0 \\ 0 & k_{n_p}(\mathbf{e}_2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\ &= \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} k_p^{11} & 0 \\ 0 & k_p^{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \end{aligned}$$

So by definition, k_p^{11} and k_p^{22} are the principal curvatures $k_{\max_p}$ and $k_{\min_p}$.

Next, we add the normal vector $\mathbf{N}_p$ to the basis $\{\mathbf{e}_1, \mathbf{e}_2\}$, and obtain the orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{N}_p\}$ in three dimensional space. Thus, we can rewrite k_p as

$$k_{n_p}(\mathbf{v}) = \begin{bmatrix} n & v_1 & v_2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_{\max_p} & 0 \\ 0 & 0 & k_{\min_p} \end{bmatrix} \begin{bmatrix} n \\ v_1 \\ v_2 \end{bmatrix}$$

$\mathbf{v} = n\mathbf{N} + v_1\mathbf{e}_1 + v_2\mathbf{e}_2$ is some vector. We want it to be the tangent vector, so we let $n = 0$ to obtain our original representation for $\mathbf{T}$. Therefore, by construction, tangent vectors to Σ at p are those vectors for which $n = 0$.

We want to generalize this expression. Notice that we can choose another set of orthonormal vectors $\{\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3\}$ as our basis and write $\mathbf{v}$ as a linear combination of these vectors. In this case

$$\mathbf{v} = u_1\mathbf{U}_1 + u_2\mathbf{U}_2 + u_3\mathbf{U}_3$$

We can substitute this expression for $k_{n_p}(\mathbf{v})$ to obtain

$$k_{n_p}(\mathbf{v}) = \mathbf{u}^T\mathbf{K}_p\mathbf{u}$$

where $\mathbf{u} = (u_1, u_2, u_3)^T$ is some vector, and $\mathbf{K}_p$ is a 3×3 symmetric matrix with 0 as one eigenvalue and $k_{\max_p}$ and $k_{\min_p}$ as the other two eigenvalues.

Because it may be very difficult to calculate $\mathbf{K}_p$ on our surface, we need to find a method by which we can estimate the curvature tensor with an eye toward our eventual goal of estimating it with discrete data. We will begin with a matrix $\mathbf{M}_p$ defined by an integral formula. $\mathbf{M}_p$ will have the same eigenvectors as $\mathbf{K}_p$, but the eigenvalues of $\mathbf{M}_p$ will be related to those of $\mathbf{K}_p$ by a linear transformation. Recall that the eigenvalues of $\mathbf{K}_p$ are the principal curvatures.

Before formally defining $\mathbf{M}_p$ we want to notice that for $-\pi \leq \theta \leq \pi$, the tangent vector $\mathbf{v}_\theta = \cos(\theta)\,\mathbf{e}_1 + \sin(\theta)\,\mathbf{e}_2$, where $\{\mathbf{e}_1, \mathbf{e}_2\}$ are the principal directions of Σ at p. Then

$$k_{n_p}(\mathbf{v}_\theta) = k_{\min_p}\cos^2\theta + k_{\max_p}\sin^2\theta$$

We will now define $\mathbf{M}_p$:

Definition 44 *The symmetric matrix* $\mathbf{M}_p$ *is defined as*

$$\mathbf{M}_p = \frac{1}{2\pi}\int_{-\pi}^{\pi} k_{n_p}(\mathbf{v}_\theta)\,\mathbf{v}_\theta\mathbf{v}_\theta^T d\theta$$

Obtaining the principal directions and curvatures reduces to diagonalizing $\mathbf{M}_p$. Because $\mathbf{v}_\theta\mathbf{v}_\theta^T$ is a rank 1 matrix for every θ, and $\mathbf{v}_\theta$ is tangent to Σ at p, $\mathbf{N}_p$ is an eigenvector corresponding to eigenvalue 0. Thus,

$$\mathbf{M}_p = \mathbf{v}_{12}^T \begin{bmatrix} m_p^{11} & m_p^{12} \\ m_p^{21} & m_p^{22} \end{bmatrix} \mathbf{v}_{12}$$

Here, $\mathbf{v}_{12} = [\mathbf{e}_1, \mathbf{e}_2]$ is the 3×2 matrix constructed by concatenating the column vectors $\mathbf{e}_1, \mathbf{e}_2$. By construction, $\mathbf{M}_p$ is symmetric, so $m_p^{12} = m_p^{21}$.

Lemma 45 *Let k_p^1, k_p^2 be the principal curvatures of S at p. Then $k_p^1 = 3m_p^{11} - m_p^{22}$ and $k_p^2 = 3m_p^{22} - m_p^{11}$.*

Proof. Notice

$$\begin{aligned} m_p^{12} &= \mathbf{e}_1^T \mathbf{M}_p \mathbf{e}_2 \\ &= \frac{k_p^1}{2\pi} \int_{-\pi}^{\pi} \cos^3\theta \sin\theta d\theta + \\ &\quad \frac{k_p^2}{2\pi} \int_{-\pi}^{\pi} \cos\theta \sin^3\theta d\theta \\ &= 0 \end{aligned}$$

since $\cos^3\theta \sin\theta$ and $\cos\theta \sin^3\theta$ are both odd functions. Thus, the two remaining eigenvectors of $\mathbf{M}_p$,other than $\mathbf{N}_p$, are the principal directions $\mathbf{e}_1, \mathbf{e}_2$. However, the corresponding eigenvalues are not the principal curvatures. We obtain

$$\begin{aligned} m_p^{11} &= \mathbf{e}_1^T \mathbf{M}_p \mathbf{e}_1 \\ &= \frac{k_p^1}{2\pi} \int_{-\pi}^{\pi} \cos^4\theta d\theta + \\ &\quad \frac{k_p^2}{2\pi} \int_{-\pi}^{\pi} \cos^2\theta \sin^2\theta d\theta \\ &= \frac{3}{8} k_p^1 + \frac{1}{8} k_p^2 \end{aligned}$$

Similarly

$$\begin{aligned} m_p^{22} &= \mathbf{e}_2^T \mathbf{M}_p \mathbf{e}_2 \\ &= \frac{k_p^1}{2\pi} \int_{-\pi}^{\pi} \cos^2\theta \sin^2\theta d\theta + \frac{k_p^2}{2\pi} \int_{-\pi}^{\pi} \sin^4\theta d\theta \\ &= \frac{3}{8} k_p^2 + \frac{1}{8} k_p^1 \end{aligned}$$

Now, we have

$$\begin{aligned} m_p^{11} &= \frac{3}{8} k_p^1 + \frac{1}{8} k_p^2 \\ m_p^{22} &= \frac{3}{8} k_p^2 + \frac{1}{8} k_p^1 \end{aligned}$$

Multiply the equation for m_p^{11} by 3, and subtract:

$$3m_p^{11} - m_p^{22} = \frac{9}{8}k_p^1 + \frac{3}{8}k_p^2 - \frac{3}{8}k_p^2 - \frac{1}{8}k_p^1 = k_p^1$$

Similarly, multiply the equation for m_p^{22} by three and subtract

$$3m_p^{22} - m_p^{11} = \frac{9}{8}k_p^2 + \frac{3}{8}k_p^1 - \frac{3}{8}k_p^1 - \frac{1}{8}k_p^2 = k_p^2$$

This completes the proof. ■

Our next task is to estimate the normal curvature $k_{n_p}(\mathbf{v})$.

Lemma 46 *The normal curvature $k_{n_p}(\mathbf{v})$ is approximated by*

$$k_{n_p}(\mathbf{v}) \approx \frac{2\mathbf{N}_p^T(q-p)}{\|q-p\|}$$

where $q \in x(s)$ is near p.

Proof. Consider some curve smooth $x(s)$ parametrized by arc length contained in S such that $x'(0) = \mathbf{v}$. We know by definition $x''(0) = k_{n_p}(\mathbf{v})\mathbf{N}_p$. We expand $x(s)$ to obtain by Taylor expansion:

$$\begin{aligned} x(s) &= x(0) + sx'(0) + \frac{s^2}{2}x''(0) + \dots \\ &= p + s\mathbf{v} + \frac{s^2}{2}k_{n_p}(\mathbf{v})\mathbf{N} + \dots \end{aligned}$$

After some algebraic manipulation, we obtain

$$2\mathbf{N}^T(x(s) - p) = k_{n_p}(\mathbf{v})s^2 + \dots$$

and

$$\|x(s) - p\| = s^2 + \dots$$

Therefore, after further manipulation and taking the limit, we obtain

$$k_{n_p}(\mathbf{v}) = \lim_{s\to 0} \frac{2\mathbf{N}_p^T(x(s)-p)}{\|x(s)-p\|}$$

Suppose $q \neq p$ is close to p on the curve $x(s)$ and $\mathbf{T}$ is the unit normalized projection of the vector $(q-p)$ onto the tangent plane. Then by replacing $x(s)$ with q, we can approximate the normal curvature as

$$k_{n_p}(\mathbf{v}) \approx \frac{2\mathbf{N}_p^T(q-p)}{\|q-p\|}$$

This completes the proof ■

This completes our discussion of the curvature tensor and normal curvature estimation in the continuous case. We will now use this information as a foundation to develop an algorithm for estimating the curvature tensor in our discrete case.

Curvature Tensor Algorithm in the Discrete Case With the proper foundation laid, our challenge is to calculate $\mathbf{M}_p$ from discrete data. We will approximate the formulas we have just established to make our calculations. We begin with a triangulated surface S representing Σ. Our goal is to estimate the principal directions and curvatures at the vertices of the triangles.

First, we need to set up some notation. A triangulated surface is represented by a pair of lists $L = \{V, F\}$ where V is the set of vertices, $V = \{v_i : 1 \leq i \leq n_v\}$, and a set of faces, $F = \{f_k : 1 \leq k \leq n_f\}$. Each face f_k consists of a set of three non-repeating vertices $\left(v_1^k, v_2^k, v_3^k\right)$. The set of vertices that share a face with v_i will be denoted by V^i, and the set of faces that contain vertex v_i will be denoted F^i. If the face f_k belongs to F^i, then f_k is said to be *incident* to v_i.

We will estimate the matrices $\mathbf{M}_{v_i}$ with a weighted sum over the neighborhood V_i. However, before we formally define $\mathbf{M}_{v_i}$ we will set up some notation used in the definition. Recall that in Section 3.1.2 we defined the normal vector at vertex v_i to be

$$\mathbf{N}_{v_i} = \frac{\sum_{f_k \in F^i} |f_k| \, \mathbf{N}_{f_k}}{\left\| \sum_{f_k \in F^i} |f_k| \, \mathbf{N}_{f_k} \right\|}$$

Now we define for each neighbor v_j of v_i, the unit normalized projection of the vector $(v_j - v_i)$.

Definition 47 *Let v_j and v_i be neighboring vertices. We define $\mathbf{T}_{ij}$ as*

$$\mathbf{T}_{ij} = \frac{\left(\mathbf{I} - \mathbf{N}_{v_i}\mathbf{N}_{v_i}^T\right)(v_i - v_j)}{\left\| \left(\mathbf{I} - \mathbf{N}_{v_i}\mathbf{N}_{v_i}^T\right)(v_i - v_j) \right\|}$$

where $\mathbf{I}$ is the 3×3 identity matrix and $\mathbf{N}_{v_i}$ is the normal vector at vertex v_i.

Now recall that we approximated the normal curvature, which we will now denote by k_{ij}, as

$$k_{ij} = \frac{2\mathbf{N}_{v_i}^T (v_j - v_i)}{\|v_j - v_i\|}$$

Our last step before formally defining $\mathbf{M}_{v_i}$ is to define our weights w_{ij}.

Definition 48 *w_{ij} be a weight assigned to the triangles incident to both vertices v_i and v_j. w_{ij} is proportional to the sum of the surface areas of these triangles. The proportionality constant is set so the sum of all weights in the neighborhood of v_j is equal to 1, i.e..*

$$\sum_{v_j \in V_j} w_{ij} = 1$$

We can now formally define our matrix $\mathbf{M}_{v_i}$.

Definition 49 *We define $\mathbf{M}_{v_i}$ as*

$$\mathbf{M}_{v_i} = \sum_{v_j \in V^i} w_{ij} k_{ij} \mathbf{T}_{ij} \mathbf{T}_{ij}^T$$

By construction, $\mathbf{N}_{v_i}$ is an eigenvector of $\mathbf{M}_{ij}$ associated with the eigenvalue of 0. We now have to calculate the other two eigenvalues. We will restrict $\mathbf{M}_{v_i}$ to the tangent plane using a householder transformation. A discussion about Householder Transformations, also called Householder Matrices, can be found in Golub [GL83]. Here, we will give the basic definition and comments taken from Golub.

Definition 50 *Let $v \in \mathbb{R}^3$ be nonzero. We define a **Householder Transformation**, or **Householder Matrix**, as the 3×3 matrix $\mathbf{P}$ of the form*

$$\mathbf{P} = \mathbf{I} - 2\frac{vv^T}{v^T v}$$

where $\mathbf{I}$ is the 3×3 identity matrix.

Remark 51 *We can construct v such that given a nonzero $x \in \mathbb{R}^3$, $\mathbf{P}x$ is a multiple of $\mathbf{E}_1 = (1,0,0)^T$. Furthermore, $\mathbf{P}x$ reflects x in the hyperplane span $\{v\}^\perp$. Notice*

$$\mathbf{P}x = \left[\mathbf{I} - \frac{2vv^T}{v^T v}\right] x = x - \frac{2v^T x}{v^T v} v$$

The requirement that $\mathbf{P}x \in span\{\mathbf{E}_1\}$ implies $v \in span\{x, \mathbf{E}_1\}$. So we can set $v = x + \alpha \mathbf{E}_1$ gives us

$$v^T x = x^T x + \alpha \mathbf{E}_1$$

and

$$v^T v = x^T x + 2\alpha \mathbf{E}_1 + \alpha^2$$

Thus

$$\mathbf{P}x = \left[1 - 2\frac{x^T x + \alpha x_1}{x^T x + 2\alpha x_1 + \alpha^2}\right] x - 2\alpha \frac{v^T x}{v^T v}\mathbf{E}_1$$

For the first coefficient of x to be zero, we set $\alpha = \pm \|x\|_2$. So, if $v = x \pm \|x\|_2 \mathbf{E}_1$ then $\mathbf{P}x = \pm \|x\|_2 \mathbf{E}_1$.

We will use this information to help us calculate the eigenvalues of $\mathbf{M}_{v_i}$ which will be related to the principal curvatures by a linear transformation. Let $\mathbf{W}_{v_i} = \frac{\mathbf{E}_1 \pm \mathbf{N}_{v_i}}{\|\mathbf{E}_1 \pm \mathbf{N}_{v_i}\|}$ choosing $\mathbf{E}_1 - \mathbf{N}_{v_i}$ if $\|\mathbf{E}_1 - \mathbf{N}_{v_i}\| > \|\mathbf{E}_1 + \mathbf{N}_{v_i}\|$. Then by construction, $\mathbf{W}_{v_i}^T \mathbf{W}_{v_i} = \|\mathbf{W}_{v_i}\| = 1$. Thus, by definition, the Householder Transformation is given by

$$\mathbf{P}_{v_i} = \mathbf{I} - 2\mathbf{W}_{v_i}\mathbf{W}_{v_i}^T$$

This matrix has its first column equal to $\pm\mathbf{N}_{v_i}$ by construction because we are working in the tangent plane. The other two columns define an orthonormal basis of the tangent space, denote them by $\mathbf{T}_1$ and $\mathbf{T}_2$. We already know $\mathbf{N}_{v_i}$ is an eigenvector of $\mathbf{M}_{v_i}$ corresponding to the eigenvalue of zero. Therefore, from linear algebra, we have

$$\mathbf{P}_{v_i}^T \mathbf{M}_{v_i} \mathbf{P}_{v_i} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & m_{v_i}^{11} & m_{v_i}^{12} \\ 0 & m_{v_i}^{21} & m_{v_i}^{22} \end{bmatrix}$$

where $m_{v_i}^{12} = m_{v_i}^{21}$. Now we can diagonalize the 2×2 minor using a Givens Rotation which results in an angle θ such that vectors

$$\mathbf{e}_1 = \cos\theta \mathbf{T}_1 - \sin\theta \mathbf{T}_2$$

$$\mathbf{e}_2 = \sin\theta \mathbf{T}_1 + \cos\theta \mathbf{T}_2$$

are the eigenvectors of $\mathbf{M}_{v_i}$. These eigenvectors are claimed to be estimates of the principal directions of the surface S at the vertex v_i.

Remark 52 *This assertion has proven to be difficult to verify mathematically, and information about Givens Rotations has been difficult to come by. However, it is probably a fairly close estimate.*

We have succeeded in approximating the normal curvature and the principal directions. Our final task is to estimate the principal curvatures k_p^1 and k_p^2. To accomplish this, we simply use the principal directions and use the linear transformation previously defined.

$$\mathbf{e}_1^T \mathbf{M}_{v_i} \mathbf{e}_1 = \lambda_1$$

$$\mathbf{e}_2^T \mathbf{M}_{v_i} \mathbf{e}_2 = \lambda_2$$

Thus, we estimate the principal curvatures at vertex v_i to be

$$k_{v_i}^1 = 3\lambda_{1_{v_i}} - \lambda_{2_{v_i}}$$

$$k_{v_i}^2 = 3\lambda_{2_{v_i}} - \lambda_{1_{v_i}}.$$

4.2.2 Methods 2 & 3

The last two methods we will discuss are found in Delingette [DEL94]. Throughout his discussion, he compares triangulation methods with his own simplex mesh method asserting that they are related to each other. We will concentrate on his estimates of the Mean and Gaussian curvatures at $p \in S$.

Mean Curvature Recall that Delingette reconstructs his surface utilizing a simplex mesh. He begins his discussion about curvature by defining a simplex angle, used in his definition of Mean curvature. However, we first want to recall our definition of a normal vector.

Definition 53 *Let S be our 2-simplex mesh representing our surface Σ. Let $p_i \in S$ be a vertex, and $(p_{i_1}, p_{i_2}, p_{i_3})$ be its three neighbors. The normal vector at vertex p_i is defined*

$$\mathbf{N}_{p_i} = \frac{(p_i - p_{i_1}) \times (p_i - p_{i_2}) + (p_{i_2} \times p_{i_3}) + (p_{i_1} \times p_{i_3})}{\|(p_i - p_{i_1}) \times (p_i - p_{i_2}) + (p_{i_2} \times p_{i_3}) + (p_{i_1} \times p_{i_3})\|}$$

We will use $\mathbf{N}_{p_i}$ to define the simplex angle.

Definition 54 *Let S_1 be the circumscribed circle at the three neighboring vertices $(p_{i_1}, p_{i_2}, p_{i_3})$. This circle is of radius r_i and of center C_i. Let S_2 be the circumscribed sphere at the four vertices $(p_i, p_{i_1}, p_{i_2}, p_{i_3})$ and let R_i and O_i be the radius and center of S_2. The **Simplex Angle** at vertex*

p_i, $\varphi_i = \angle(p_i, p_{i_1}, p_{i_2}, p_{i_3})$ *is defined as*

$$\varphi_i \in [-\pi, \pi]:$$

$$\sin(\varphi_i) = \frac{r_i}{R_i}\left(sign\left(\overrightarrow{p_i p_{i_1}} \cdot \mathbf{N}_i\right)\right) \quad and$$

$$\cos(\varphi_i) = \frac{\|O_i C_i\|}{R_i}\left(sign\left(\overrightarrow{O_i C_i} \cdot \mathbf{N}_i\right)\right)$$

Here, $\|O_i C_i\|$ is the distance of the plane $(p_{i_1}, p_{i_2}, p_{i_3})$ from the center of the sphere, see figure 6 adapted from [DEL94]. Figure 7 shows that the simplex angle can be thought of as a planar angle. We will use $\sin(\varphi_i)$ to define the mean curvature.

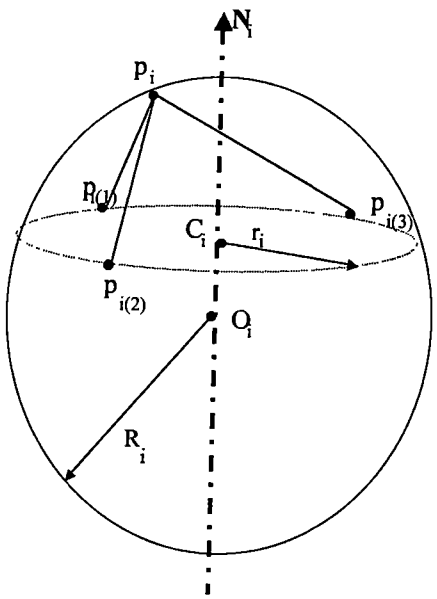

Figure 6: Local tetrahedron and the associated circumscribed sphere and circle

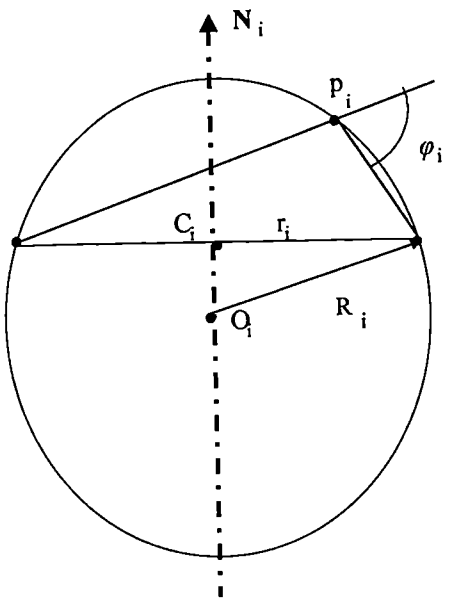

Figure 7: Cross Section of 6 through plane defined by p_i, O_i, and C_i

Definition 55 *Let p_i be the vertex of a simplex mesh S. If φ_i and r_i are its simplex angle and the radius of the circumscribed circle, we define the* ***Mean Curvature*** *H_i at p_i as*

$$H_i = \frac{\sin(\varphi_i)}{r_i}$$

Lemma 56 *The absolute value of the mean curvature is the inverse of the radius of the circumscribed sphere at p_i*

$$|H_i| = \frac{1}{R_i}$$

Proof. This is straightforward. First, recall that

$$\begin{aligned}\sin(\varphi_i) &= \frac{r_i}{R_i}\left(sign\left(\overrightarrow{p_i p_{i_1}} \cdot \mathbf{N}_i\right)\right) \\ &= \pm\frac{r_i}{R_i}\end{aligned}$$

Then

$$
\begin{aligned}
|H_i| &= \left| \frac{\sin(\varphi_i)}{r_i} \right| \\
&= \left| \frac{1}{r_i} \left(\frac{r_i}{R_i} \right) \right| \\
&= \left| \frac{1}{R_i} \right|
\end{aligned}
$$

This completes the proof. Figure 8 is an illustration of this method of estimating Mean Curvature (adapted from [DEL94]) ■

We obtain the correct value of the mean curvature of a sphere, so our definition is mathematically equivalent except for a plane. Notice that we cannot classify a point on a surface with the mean curvature as our only parameter. Consider two points $p \in S$ and $q \in S$ such that $p \neq q$ where $H_p = H_q$. With mean curvature as our only descriptive parameter, we would have to assume that the surface at p "looks like" a surface at q. However, this may not be the case. For example, suppose $H_p = H_q = 0$ and that the surface at p is a plane. With mean curvature as our only parameter, we are forced to assume the surface at q is a plane. But, minimal surfaces also have $H = 0$ which are certainly are not planar. To accomodate this shortcoming, other metric parameters are introduced to describe the location of a vertex with respect to its three neighbors.

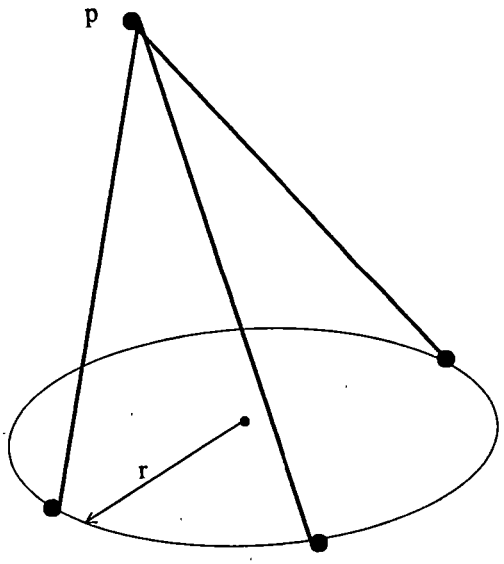

Figure 8: Mean Curvature on a 2-Simplex Mesh

We will now move onto our discussion about Gaussian Curvature.

Gaussian Curvature Finally, we want to discuss Gaussian curvature on a triangulation. This approach is also discussed in Delingette. Consider a point $p \in S$. The discrete Gaussian curvature on a triangulation is estimated through the *spherical excess.*

Definition 57 *Let p_i be a vertex of a triangulation. Let $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$ be the neighbors of p_i connected by edges $e_{i_1}, e_{i_2}, \ldots, e_{i_n}$ where e_{i_1} connects p_i to p_{i_1}, e_{i_2} connects p_i to p_{i_2} etc. Let $\theta_1 = \angle(e_{i_1}, e_{i_2})$, $\theta_2 = \angle(e_{i_2}, e_{i_3})$, $\blacksquare$, $\theta_n = \angle(e_{i_n}, e_{i_1})$. Let $\theta = 2\pi - \sum_{j=1}^{n} \theta_j$. Finally, let $A_{i_1}, A_{i_2}, \ldots, A_{i_n}$ be the areas of the triangles formed by the neighboring vertices of p_i, and let $A = \sum_{j=1}^{n} A_j$. Then we define the **Gaussian curvature**, K, to be*

$$K = \frac{\theta}{A}$$

This definition attempts to make use of the Local Gauss Bonnet theorem, which is stated as follows from DoCarmo [DC76]:

Theorem 58 ***(Local Gauss Bonnet Theorem)*** *Assume $\mathbf{x} : U \to S$ be a parametrization of an oriented surface S. Let $R \subset \mathbf{x}(U)$ be a simple region of S and $\alpha : \mathbf{I} \to S$ be such α is piecewise smooth and $\partial R = \alpha \mathbf{I}$, $\mathbf{I} = [0, 1]$. Assume α is positively oriented, parametrized by arc length s, and let $\alpha(s_0), \ldots, \alpha(s_k)$ and $\theta_0, \ldots, \theta_k$ be the vertices and external angles of α. Then*

$$\sum_{i=0}^{k} \int_{s_i}^{s_{i+1}} k_g(s)\, ds + \int\int_R K d\sigma + \sum_{i=0}^{k} \theta_i = 2\pi$$

where $k_g(s)$ is the geodesic curvature of the regular arcs of α and K is the Gaussian curvature of S.

Remark 59 ***(Explanation of Gaussian Curvature Estimation on a Triangulation)*** *We will alter the notation to make this explanation a bit simpler. We will let θ_i be denoted by β_{i_2}. Thus, β_{i_2} is now the vertex angle. Denote the other two angles in each triangle as β_{i_1} and β_{i_3}. Thus, the exterior angle, denoted by θ_i in the Gauss Bonnet theorem, has measure*

$$\theta_i = \pi - \left(\beta_{i_3} + \beta_{(i+1)_1}\right)$$

So the sum of the exterior angles is:

$$\begin{aligned}\sum_{i=0}^{k}\theta_i &= \pi-(\beta_{1_3}+\beta_{1_1})+\pi-(\beta_{2_3}+\beta_{3_1})+\ldots+\pi-(\beta_{n-1_3}+\beta_{n_1})+\pi-(\beta_{n_3}+\beta_{1_1})\\ &= (\pi-\beta_{1_1}-\beta_{1_3})+\ldots+(\pi-\beta_{n_1}-\beta_{n_3})\\ &= \beta_{1_2}+\beta_{2_2}+\ldots+\beta_{n_2}\\ &= \sum_{i=1}^{n}\beta_{i_2}\end{aligned}$$

Thus, the sum of the exterior angles is simply the sum of the vertex angles at vertex p. By DoCarmo [DC76] and the preceding discussion,

$$\begin{aligned}2\pi &= \sum_{i=0}^{k}\int_{s_i}^{s+1}k_g(s)\,ds+\int\int_R K\,d\sigma+\sum_{i=1}^{k}\theta_i\\ &= \sum_{i=0}^{k}\int_{s_i}^{s+1}k_g(s)\,ds+\int\int_R K\,dA+\sum_{i=1}^{k}\beta_{i_2}\end{aligned}$$

Assuming K is constant near the vertex p,

$$2\pi = \sum_{i=0}^{k}\int_{s_i}^{s+1}k_g(s)\,ds+K\int\int_R dA+\sum_{i=1}^{k}\beta_{i_2}$$

In our case, we are working with individual planar triangles, so the edges are straight which means that $k_g(s)=0$. Therefore, we have

$$K=\frac{2\pi-\sum_{i=1}^{k}\beta_{i_2}}{\int\int_R dA}$$

Finally $\int\int_R dA=\sum_{i=1}^{n}A_i$ because our region R is subdivided into a finite number of triangles (see figure 9). Hence

$$\begin{aligned}K &= \frac{2\pi-\sum_{i=1}^{k}\beta_{i_2}}{\sum_{i=1}^{k}A_i}\\ &= \frac{\theta}{A}\end{aligned}$$

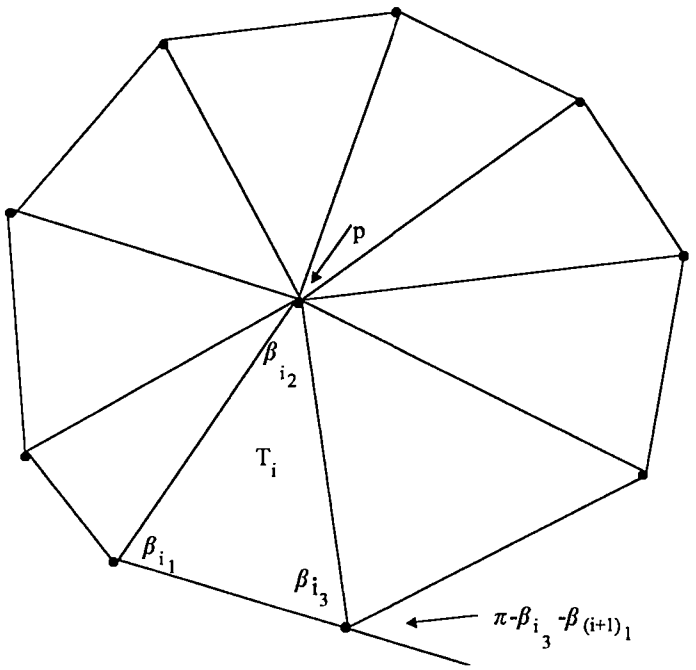

Figure 9: Gaussian Curvature on a Triangulation

4.3 Conclusion:

As we have seen, curvature can be estimated in several different ways. We have estimated curvature directly from the discrete data and from functions which approximate the data. This has by no means been an exhaustive study of every method for estimating curvature, this has simply been an overview of a few of them. One issue we want to address is which approach is most reliable and efficient. In both approaches, noise will play a role in the accuracy of each estimation.

1. Curvature is a second order quantity. Thus, it will be highly sensitive to noise.

2. Least squares formulas are highly affected by small changes in distances between data points. Thus, function estimation is highly sensitive to noise.

Because we do not take derivatives on a triangulation or simplex mesh, noise will be less significant in these approaches because we can more easily adjust our triangulation to the effects of noise. However, we do not have this luxury when we estimate functions. The effects of noise cannot be filtered out.

Another difficulty with calculating curvature from approximating functions is at the intersection of two functions. These intersections may not result in a smooth function. As a result, smoothing may be required so that we can create a continuously smooth function. Doing this will further alter our surface approximation and lead to further noise contamination. Unfortunately, Flynn [FLN89] asserts that this smoothing is necessary in order to obtain any decent estimation.

Finally, the size of the neighborhood we are approximating by smooth functions also affects the accuracy of curvature estimation. According to Flynn, we cannot obtain a reliable estimate of curvature with very small neighborhoods, but we cannot obtain a reliable continuous function estimating the data with larger neighborhoods because of the problem with patching functions. In experiments, Flynn [FLN89] could not obtain better than 10% accuracy of the curvature values.

We also discussed curvature estimation directly from the triangulation or simplex mesh. These approaches are also sensitive to noise, although the problem with the noise is not compounded further by function estimation. As a result, the estimates are a bit more accurate than the function-based approaches [FLN89]. They also utilize the strength of computers to handle large amounts of discrete data and make calculations from it. With these approaches, smoothing is not necessary because we are not attempting to calculate second derivatives.

We also considered different kinds of curvature. Some are more useful in surface recreation than others. As we discussed, different points on a surface may have the same Mean or Gaussian Curvatures, but the surface at those points may be very different. Thus, other aspects of the surface at those points must be evaluated to create an accurate reproduction. To reduce the number of parameters required to reproduce the surface, principal curvatures and their corresponding directions are usually chosen.

Curvature values are used in many aspects of computer visualization. They are used directly in surface reconstruction when function estimation is used to recreate the surface, but they are not used in surface reconstruction directly when a triangulation method is used. In one case, we

will see that curvature estimation is used to determine sampling criterion. However, curvature estimation is used extensively to partition the resulting surface into its constituent parts, called segmentation[Wu97][MN99] [RS93][MD00]. We do not discuss segmentation in this thesis.

5 SURFACE RECONSTRUCTION

In Chapter 3 we discussed methods by which normal vectors are calculated from a point cloud. These vectors are important in most surface reconstruction algorithms because normal vectors determine how the reconstructed surface will be oriented. If the normal vectors do not adequately represent the normal directions of the original surface, the reconstruction will be inaccurate. We have already discussed each of these methods in chapter 3 except 1. A single method, the Ball Pivoting Algorithm [BN99], is not dependent on the method of calculating the normal vector nor on any other differential geometry. It is a completely unique approach, but is included here because it produced very accurate results by utilizing a very simple idea.

We will begin with a method that uses the point cloud to approximate smooth functions. The properties of these functions will be used to partition the point cloud into volumetric primitives to recreate the surface. The resulting surface will be the union of these volumetric primitives. We will then discuss methods that produce a triangulation of the surface. In either case, the goal is to recreate a surface, S, homeomorphic to the original surface Σ. Each discussion will begin with an overview of the method being explored followed by its details. These methods are not always mathematically precise; however, these methods are discussed because it is the purpose of this thesis to explore the methods by which engineers use mathematics to create surfaces.

5.1 Smooth Function Approximation

Because surfaces can be represented and studied via Differential Geometry, perhaps the most obvious method by which we can create the surface is to transform the data from the range camera into a form which allows us to apply differential geometry directly. Since all surfaces can be described locally by a smooth function, or parametrization of the surface, this is an obvious, if not the most efficient or accurate, method to recreate a surface. We will take a look at one method which uses this general philosophy.

This method is proposed by Ferrie [FR98]. The primary goal of this particular method is to create a surface which is based only on the primary components of the surface for the purposes of allowing robots to recognize that something is in their path or determine a way to manipulate the

object. For example, if a maniquin was in its path, the head would be represented by an ellipsoid, the torso, two arms and legs would all be represented as cylinders as well as the ten fingers. No details of the maniquin, such as eyes, would appear in the recreation.

The robot collects data about the surface via a range camera as it approaches the object. It obtains different samples at different times, so a method must be developed to ensure that we do not reproduce the same components of the object more than once. Because the goal of this method is not to produce a detailed replica of the surface, but only a structural representation of it, it only identifies basic structural properties such as the principal axes of the object and the relative size and shape of the different components. Because the robot cannot obtain data over the entire surface, we assume the object is symmetrical. For example, if the front of the surface is determined to be a cylinder, the back of the surface is assumed to complete the cylinder.

As previously stated, differentiable surfaces can be represented by continuous functions. A group of the surfaces we are concerned with is called *volumetric primitives* and was defined in chapter 3. Ferrie utilizes a subgroup of these primitives to produce a surface as the union of these primitives. The difficulty arises from the nature of the data we have to work with. Our data is discrete, and functions require continuous data. To obtain a close approximation to discrete data, we can only approximate very small neighborhoods by these smooth function. We use these locally defined functions to estimate normal vectors and principal curvatures, discussed in previous chapters. After this is accomplished, will trace through these estimates of principal curvatures to locate large discontinuities in the curvature values. These discontinuities will determine partitioning of the point cloud into larger neighborhoods. The volumetric primitives are fitted to these larger neighborhoods and are determined by the ratios of the principal curvatures.

3-dimensional images can create problems in that no view of the surface, or data representing the surface, can give us a complete picture. For instance, any view of a sphere can only reveal, at most, a partial hemisphere. Without having to take additional data, we can extrapolate from this surface patch that it is a sphere and complete the unseen portion. This leads to the guiding principal:

> If we can infer the larger geometry of a patch by extrapolating its properties, we can

instantiate, or identify and describe, the volumetric primitive of which it is a part.

This will greatly reduce the amount of time needed to create an image, which is important if the object is in the path of a moving robot. Furthermore, a robot will not have the ability to take multiple passes with its camera, or eyes, at different positions around the object, thus, it is necessary to "make the most" out of the little information we get. The process is broken down into three steps:

1. **Parts Decomposition:** We partition the object into a set of regions, or patches, corresponding to a single part.
2. **Primitive Instantiation:** We assume each partition is a sample of a larger primitive and estimate the parameters of the primitive from the sample.
3. **Resolution:** Eliminate multiple instantiations.

During the parts decomposition step, we want to find a partitioning of the data which is as natural as possible. We have PC representing the surface Σ of the form $(x, y, f(x, y))$. We define a surface patch as a partition of PC.

Definition 60 *We define a* ***surface patch*** *as a subset of* PC.

$$PC = \cup_{i=1}^{n} PC_i \text{ such that } PC_i \subset PC \text{ and } PC_i \cap PC_j = \emptyset$$
$$i, j = 1, \ldots, n;\ i \neq j$$

We now let Γ represent the set of volumetric primitives. From the set of partitions, we want to infer a volumetric rendition of S.

Definition 61 *We define our volumetric rendition of our surface recreation from the* PC *as follows*

$$S \approx V = \cup_{j=1}^{m} V_j : V_j \in \Gamma$$

The challenge now become how to partition our point cloud. We utilize an idea from physical psychology, [MN99], which conjectures that the human vision system partitions objects based on areas of undefined normal curvature. The idea is that when two objects come together, they form a

contour which corresponds to a discontinuity in the normal direction which means that the normal curvature is undefined. However, we can smooth out the contour so that the normal curvature is defined at these points. Once smoothed, the values of the maximum and minimum curvature, $k_{\min}$ and $k_{\max}$, will be a local extremaa at these points of intersection. Thus, we can decompose the set of points according to the areas of local maxima of principal curvatures. We want to find a method by which we can use our estimates the principal curvatures. We accomplish this by creating an approximating function in a local neighborhood. We call this approximating function $\overline{f}$ with the coordinate system centered at some $p \in PC$. We define it as described earlier

$$\overline{f} = a_1u^3 + a_2u^2 + a_3u + a_4u^2v^2 + a_5u^2v + a_6uv^2 + a_7uv + a_8v^3 + a_9v^2 + a_{10}v + a_{11}$$

Recall that we found the principal curvatures, $k_{\min_p}$ and $k_{\max_p}$, evaluated at p to be

$$k_{\max_p} = a_2 + a_9 + \sqrt{(a_2 - a_9)^2 + a_7}$$
$$k_{\min_p} = a_2 + a_9 - \sqrt{(a_2 - a_9)^2 + a_7}$$

We now proceed with the next step in our algorithm.

Remark 62 *$\overline{f}$ is no longer used throughout the rest of this algorithm. Its only purpose was to estimate local values of principal curvatures. Notice that any algorithm to estimate these values could have been used since because the main idea behind the algorithm is in partitioning the point cloud into parts that represent the major components of the original surface.*

One might naturally ask why we go through the trouble of approximating smooth functions to the data if we are only going to use them for a specfic calculation and not in the rest of the algorithm. Why not use some other curvature calculation already proven? According to Ferrie, this calculation may be faster than curvature approximation based only on the discrete data. We can also speculate that because Ferrie is using volumetric primitives to recreate the surface, he wanted to be consistent in curvature approximation by using smooth functions representing volumetric primitives. However, this is not certain because he does not address this in his paper.

Primitive instantiation is the process by which we determine the primitives corresponding to the partitions of PC. Table 1 shows the signs of the principal curvatures corresponding to the various primitives.

Table 1: Generic Surface Classification

Primitive	**Signs** $(k_{\max}, k_{\min})$
Sphere	$(+,+),(-,-)$
Ellipsoid	$(+,+),(-,-)$
Cylinder	$(+,0),(-,0)$
Hyperboloid	$(+,-)$
Planar	$(0,0)$

We differentiate the Sphere and Ellipsoid by their ratios $\frac{k_{\max}}{k_{\min}}$. If $\frac{k_{\max}}{k_{\min}} \approx 1$, the surface is a sphere. Otherwise, it is an ellipsoid. The difficulty occurs in regions which are partially hyperbolic and partially elliptic. These transitions occur through inflections in the Gaussian curvature $K = k_{\min} * k_{\max}$. These inflection points are called parabolic and have Gaussian curvature as defined as in chapter 2 to be $K = 0$ and either $k_{\min} \neq 0$ or $k_{\max} \neq 0$. Thus, to partition PC, we will evaluate the Gaussian curvature. We parametrize $K(u,v) = k_{\max}(u,v) \times k_{\min}(u,v)$ via a first order approximation of the form $K(u,v) = au + bv + c$ and evaluate whether or not there is a zero crossing in the vicinity of p. We also want to locate the extremal points.

Definition 63 *We define* ***extremal points*** *as those points p_i as extremal points if $\left|k_{\min_p}\right|$ or $\left|k_{\max_p}\right|$ are local extrema.*

To accomplish this, we assume that the magnitudes of curvature values are much higher in the vicinity of part intersections than those of the surrounding neighborhood. We compute histograms of $|k_{\min}|$ and $|k_{\max}|$ to locate these points in a local neighborhood. The local extrema within the region will show up as a cluster in the histogram and can be identified as a peak at the higher magnitudes that exceed a threshold. If there are no discernible peaks, we assume there are no extremal points in the region. Examples of a histogram appear on the following page in figures 10 and 11 adapted from [FR98].

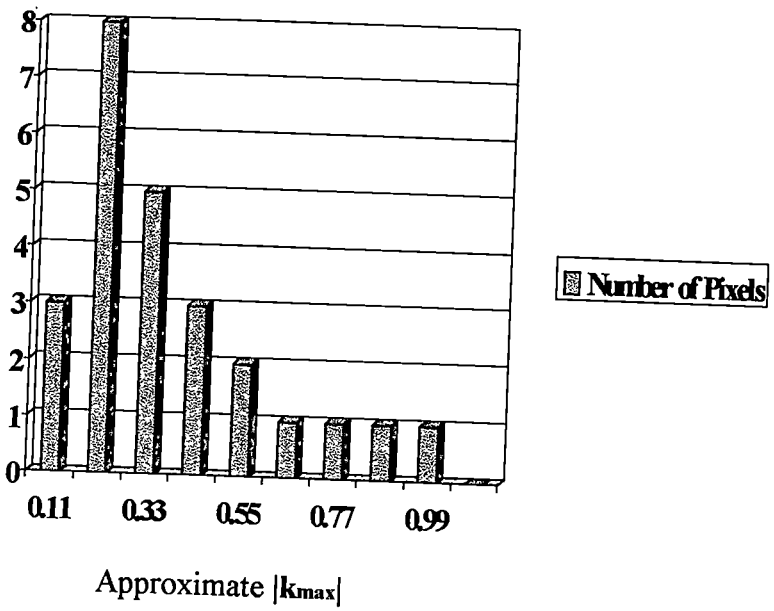

Figure 10: Histogram of Maximum Principal Curvature Magnitudes of an Owl

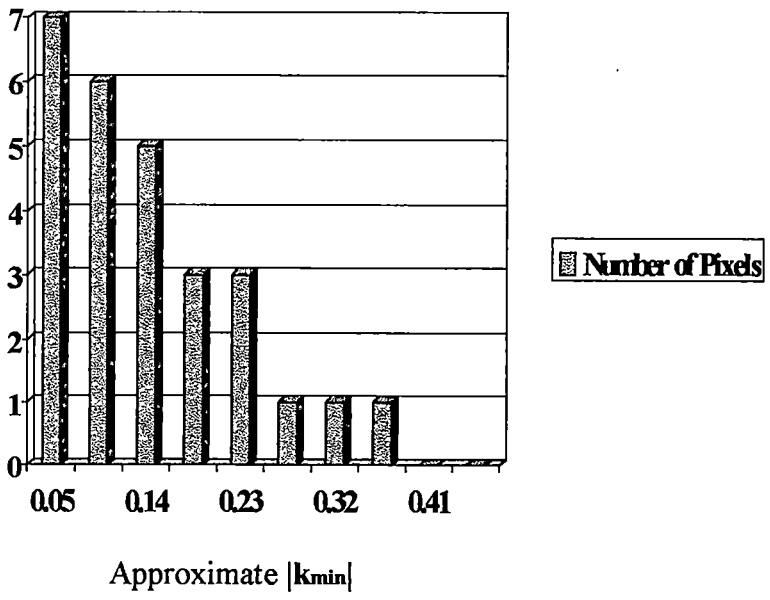

Figure 11: Histogram of Minimum Principal Curvature Magnitude of an Owl

These feature points are now grouped into extremal and parabolic contours. Contours formed by extremal points are used to partition the PC into a set of "patches" PC_i, and contours formed by parabolic points are used to subdivide PC_i into elliptic and non-elliptic segments.

The final step in this process is resolution, or making the final inferences and eliminating multiple instantiations of primitives. Consider different views of an object. These views are taken at times $t_j; j = 1, \ldots, n$ is the j^{th} view. First, each view is mapped into the same frame of reference via linear transformation which maps coordinates of each surface patch at time j, $PC_i^{t_j}$, into a common frame of reference, call it W and infer a volumetric primitive of PC_i using principal curvatures. Recall that the principal curvatures were found from the local approximating function $\overline{f}$ about $p \in PC$. Thus, there is no approximating function for PC_i from which to directly calculate the curvatures. We address this by defining the global Gaussian curvature on PC_i.

Definition 64 *We define the* ***Gaussian Curvature*** *on PC_i as follows:*

$$\overline{K} = \frac{\sum_{PC_i} \overline{k}_{\min}(x,y)}{\sum_{PC_i} \overline{k}_{\max}(x,y)}$$

where $\overline{k}_{\min}$ and $\overline{k}_{\max}$ are the mean values of $k_{\min}$ and $k_{\max}$ at each $p \in PC_i$.

This is clearly not the usual definition of Gaussian Curvature (recall that Gaussian Curvature is defined mathematically as $k_{\min_p} \times k_{\max_p}$) and is mathematically misleading. This definition is used because we want to be able to utilize the above table to determine our primitive. For a cylindrical surface $\overline{K}$ will be very small; spheres and ellipsoids will have $\overline{K} > 0$ with spheres having $\overline{K} \approx 1$; hyperbolic surfaces will have $\overline{K} < 0$.

We now have partitioned PC and characterized each partition as a corresponding primitive. To parametrize the surface, we want to minimize

$$|g_i(x,y,z) - S_i(x,y,z)|$$

where $g_i(x,y,z)$ is the parametric representation of the instantiated model, or the primitive S_i was identified as. We can either minimize this function directly or use constraints obtained during instantiation. The only example using this process given is when S_i is a sphere. Suppose $p \in S_i$, S_i is identified as a sphere and $\mathbf{N}_p$ is the normal vector to the sphere at p. Then we find $g_i(x,y,z)$

by evaluating the center.

$$r_c = p - \frac{1}{k}\mathbf{N}_p$$

Note that k is the value of the second fundamental form at p and is the same for all directions. This is a common formula and can be found in Millman [MIL77]. The formulas for the other primitives are much more complicated and are not discussed.

Second, overlapping surface patches are eliminated. We impose a uniqueness constraint so that each volumetric primitive is instantiated no more than once in a sequence of views which can be ensured by requiring that no two primitives with the same volumetric description can occupy the same position in space. We accomplish this by comparing parametric descriptions. We will define two similarity functions for ellipsoids and cylinders.

Definition 65 *Define the similarity function for ellipsoids to be*

$$\begin{aligned} D_e(V_i, V_i^{'}) = & \alpha_e \sqrt{(X_c - X_c^{'})^2 + (Y_c - Y_c^{'})^2 + (Z_c - Z_c^{'})^2} \\ & + \beta_e \left(\left|\theta_x - \theta_x^{'}\right| + \left|\theta_y - \theta_y^{'}\right| + \left|\theta_z - \theta_z^{'}\right| \right) \\ & + \gamma_e \left(|a - a'| + |b - b'| + |c - c'| \right) \end{aligned}$$

where (X_c, Y_c, Z_c) *and* $\left(X_c^{'}, Y_c^{'}, Z_c^{'}\right)$ *are the centroids,* $(\theta_x, \theta_y, \theta_z)$ *and* $\left(\theta_x^{'}, \theta_y^{'}, \theta_z^{'}\right)$ *are the direction cosines of the principal axes, and* (a, b, c) *and* (a', b', c') *are the axis lengths of elliptical volumes* V_i *and* $V_i^{'}$ *respectively.* α, β, *and* γ *are weighting parameters which reflect the certainty of the information.*

Definition 66 *Define the similarity function for cylinders to be*

$$\begin{aligned} D_c(V_i, V_i^{'}) = & \alpha_e \sqrt{(X_1 - X_1^{'})^2 + (Y_1 - Y_1^{'})^2 + (Z_1 - Z_1^{'})^2} \\ & + \beta_c \sqrt{(X_2 - X_2^{'})^2 + (Y_2 - Y_2^{'})^2 + (Z_2 - Z_2^{'})^2} \\ & + \gamma_e \left(|R - R'| \right) \end{aligned}$$

where (X_1, Y_1, Z_1), (X_2, Y_2, Z_2) *and* $\left(X_1^{'}, Y_1^{'}, Z_1^{'}\right)$, $\left(X_2^{'}, Y_2^{'}, Z_2^{'}\right)$ *are the endpoints and* R *and* R' *are the radii of cylindrical volumes* V_i *and* $V_i^{'}$ *respectively.* α, β, *and* γ *are weighting parameters which reflect the certainty of the information.*

In both of these definitions, α, β, and γ are determined experimentally. Notice that the parameters used in these definitions will be unique for each primitive in a particular area. If the values of D_c and D_e fall below a user defined threshold, we determine they are multiple instantiations. We will eliminate one of them from our reproduction, and the union of the remaining primitives represent our surface. Figure 12 is an example of this process taken from [FR98].

5.2 Triangulation

In general, triangulation algorithms take points in the point cloud and determine how to connect them in such a way as to approximate the original surface via a set of small, connected triangles. In the first two methods, the general idea is that if you are close enough to a given point on a smooth surface, those points will be near the points in the tangent plane. The primary challenge is developing reliable methods to determine which points should be vertices of the same triangle. By restricting ourselves to considering only points in small neighborhoods we will reduce computational time and the chances of error. Points are considered vertices of the same triangle via a distance evaluation.

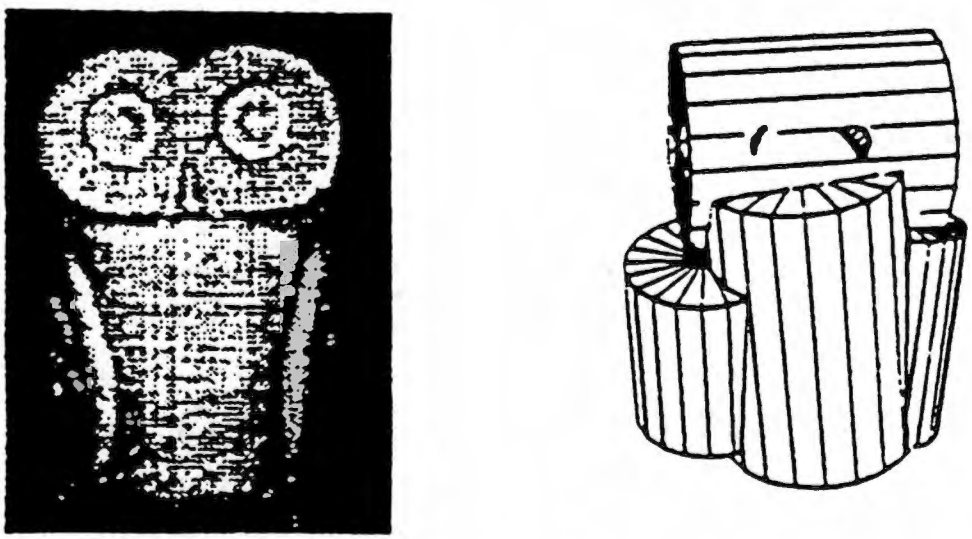

Figure 12: A Recreation of an Owl

The third method we will study is a departure from these general approaches in that the tangent plane is not used. Basically, the idea is to roll a ball across the point cloud to determine the triangulation. The ball begins in contact with three points, called a seed triangle, and begins rolling over one of the edges until it hits another data point. This point creates another triangle. This process continues until all the points in the point cloud are touched by this ball.

Before we begin, we need to state a definition central to assumptions in these algorithms.

Definition 67 *Let $p \in PC$. A point cloud, PC, is considered δ-dense if the distance between p and the nearest element $q \in PC$ is no larger than some constant δ. We denote this by $d(p, PC) \leq \delta$. A point cloud, PC, is considered ε-noisy if the error in the point cloud is no more than some constant ε.*

In each of these algorithms, we assume the point cloud is δ-dense and ε-noisy. Notice that these assumptions imply that $d(p, PC) \leq \delta + \varepsilon$, so the distance between a point p and its nearest point $q \in PC$ is less than or equal to $\delta + \varepsilon$. We will begin with basic triangulation approaches.

5.2.1 Method 1

Hoppe et.al. [HOP92] produces a triangulation using the normal vector calculation we discussed in section 3.1.1. The following assumptions are made about both the surface Σ and the point cloud:

1. The point cloud is PC is δ-dense and ε-noisy.
2. A surface can be locally approximated by tangent planes.
3. The original surface Σ is known

This algorithm begins by defining a tangent plane $\mathbf{T}(p)$ for each $p \in PC$. Then, using what is called the *center* of the tangent plane we consistently orient them. An initial triangulation is determined by connecting these centers via a *Riemannian Graph.* To refine the triangulation, we choose an arbitrary point $P \in \mathbb{R}^3$ and project it to the nearest tangent plane. Next, we define the signed distance function, $f : D \to \mathbb{R}$ where $D \subset \mathbb{R}^3$ is a region near the PC. f represents the signed geometric distance from P to the surface S. Our final surface is determined by the zero set of f,

denoted by $Z(f)$. $Z(f)$ is defined by all points $P \in \mathbb{R}^3$ such that $f(P) = 0$. To create the final triangulation, we will use a modified marching cubes algorithm to connect the points in the zero set of f.

Again, the central idea behind this algorithm is the signed distance function. This function takes as input a point $P \in \mathbb{R}^3$ and computes the signed distance from P to the surface S approximated locally by tangent planes associated with each $p_i \in PC$. Before we give the formal definition of f, we need to define the tangent plane at each $p_i \in PC$ and its orientation.

At each point $p \in PC$ we define a tangent plane $\mathrm{T}(p)$ by utilizing $nbhd(p)$. The user determines this neighborhood by specifying the number of nearest points to p to be considered. Let c_p be the centroid of $nbhd(p)$ and $\mathbf{N}_p$ be the normal vector to p defined in Section 3.1.1. $\mathrm{T}(p)$ is defined in the usual way with the point c_p and the normal vector $\mathbf{N}_p$.

Definition 68 *The **Tangent Plane** containing* $c_p = (c_{p_1}, c_{p_2}, c_{p_3})$ *with normal vector* $\mathbf{N}_p = (n_{p_1}, n_{p_2}, n_{p_3})$ *is defined by*

$$0 = \langle c - c_p, \mathbf{N}_p \rangle$$

$$= n_{p_1}(c_1 - c_{p_1}) + n_{p_2}(c_2 - c_{p_2}) + n_{p_3}(c_3 - c_{p_3})$$

for $c = (c_1, c_2, c_3)$. *We will denote this plane by* $\mathrm{T}(p) = (c_p, \mathbf{N}_p)$ *and we call* c_p *the center of* $\mathrm{T}(p)$

We concern ourselves now with consistent plane orientation, making sure that all the normal vectors are consistently oriented. Remember, we chose the normal vector at p to be $\mathbf{N}_p = \pm\mathbf{e}_1$, the eigenvector corresponding to the smallest eigenvalue of the covariance matrix. The sign of $\mathbf{N}_p$ is chosen so that the nearby tangent planes are consistently oriented. Thus, we must discuss how to choose either $+\mathbf{e}_1$ or $-\mathbf{e}_1$. After this is done, we can define our distance function.

Consider $p_i \in PC$ and $p_j \in PC$ where $p_i \neq p_j$. Suppose they are "geometrically close." Ideally, they will be geometrically close if $\mathbf{N}_{p_i}$ and $\mathbf{N}_{p_j}$ are nearly parallel. This assumption makes sense since we assume the surface is smooth and contains no radical changes. Thus, $\langle \mathbf{N}_{p_i}, \mathbf{N}_{p_j} \rangle \approx \pm 1$. If the planes are consistently oriented, $\langle \mathbf{N}_{p_i}, \mathbf{N}_{p_j} \rangle \approx 1$. If $\langle \mathbf{N}_{p_i}, \mathbf{N}_{p_j} \rangle \approx -1$, then either $\mathbf{N}_{p_i}$ or $\mathbf{N}_{p_j}$ should be flipped to orient them consistently. Finding a consistent global orientation turns out to be a difficult problem because the condition should hold between all pairs of sufficiently close data points.

Definition 69 *Two points c_i and c_j are called* ***sufficiently close*** *if $c_i \in nbhd(c_j)$ or if $c_j \in nbhd(c_i)$*

To accomplish this, we assign to each tangent plane $\mathbf{T}(p_i)$ a "node," or a point, denoted by N_i. Our goal is to connect these nodes to each other to enrich the data. N_i and N_j will be connected by edge (i,j) if c_{p_i} and c_{p_j} are sufficiently close. The cost on edge (i,j) will encode the degree to which N_i and N_j are consistently oriented and is taken to be $\mathbf{N}_{p_i} \cdot \mathbf{N}_{p_j}$. The goal is to minimize this cost. We call this a *Riemannian Graph* and it encodes the geometric proximity of the tangent plane centers, see figure 13 from [HOP92].

Now we propagate the normal vectors to achieve global orientation. We assign to each edge (i,j) the cost $1 - \left|\mathbf{N}_{p_i} \cdot \mathbf{N}_{p_j}\right| > 0$. If the normal vectors are nearly parallel, the cost is very small. We proceed along each neighborhood and calculate the cost. To assign orientation to an initial plane, the unit normal of the plane whose center has the largest z-coordinate is forced to point toward the $+z$ axis. Traverse the $nbhd(c_{p_i})$ and if the current plane $\mathbf{T}(p_i)$ has been assigned orientation $\mathbf{N}_{p_i}$ and $\mathbf{T}(p_j)$ is the next plane to be visited, then $\mathbf{N}_{p_j}$ is replaced with $-\mathbf{N}_{p_j}$ if $\mathbf{N}_{p_i} \cdot \mathbf{N}_{p_j} < 0$.

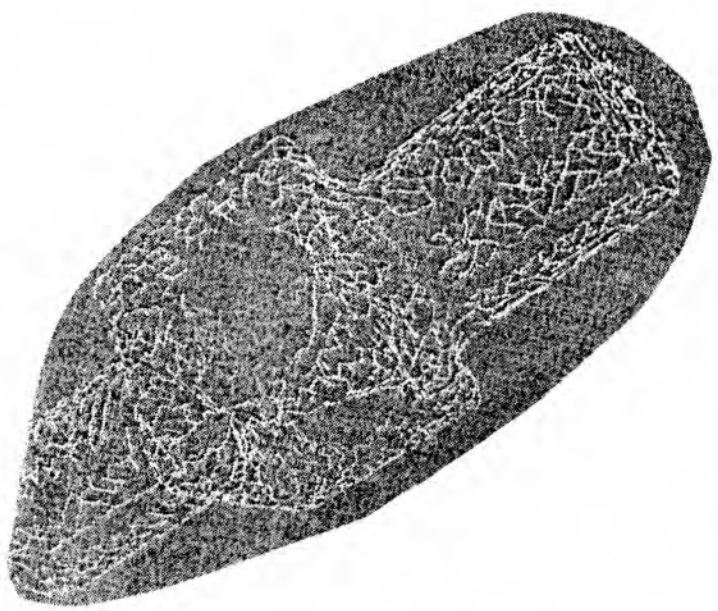

Figure 13: Riemannian Graph over Tangent Plane Centers

We can now begin defining the signed distance function. First, we take an arbitrary point $P \in \mathbb{R}^3$ close to the data set , find the tangent plane $\mathbf{T}(p_i)$ whose center c_i is closest to P. This plane is a local linear approximation of Σ, so we take the signed distance $f(P)$ to be the signed distance between P and its projection z onto the tangent plane.

Definition 70 *The **Projection** z of arbitrary point $P \in \mathbb{R}^3$ onto the tangent plane $\mathbf{T}(p_i)$ is defined as*

$$z = c_{p_i} - ((P - c_{p_i}) \cdot \mathbf{N}_{p_i})\,\mathbf{N}_{p_i}$$

Definition 71 *The **Signed Distance Function** is defined as*

$$f(P) = dist_{\mathbf{T}_{p_i}}(P) = (P - c_{p_i}) \cdot \mathbf{N}_{p_i}$$

for $P \in \mathbb{R}^3$

We don't want to test points near the data if those data points are a result of errors during the collection of data. To accomplish this, we calculate the Euclidean distance from z to the point cloud. Recall that we are assuming that PC is δ-dense and ε-noisy. Thus, z will not be considered a point of S if $d(z, PC) > \varepsilon + \delta$. The following algorithm describes this process.

Algorithm 72 *The algorithm for the signed distance function*

1. $i \leftarrow$ *index of the tangent plane whose center is closest to* P
2. $z \leftarrow c_{p_i} - ((P - c_{p_i}) \cdot \mathbf{N}_{p_i})\,\mathbf{N}_{p_i}$ $\{z$ *is the projection of* P *onto* $T(p_i)\}$
3. ***if*** $d(z, PC) \leq \delta + \varepsilon$ ***then*** $f(P) \leftarrow (P - c_{p_i}) \cdot \mathbf{N}_{p_i}$ $\{= \pm \|P - z\|\}$
4. ***else*** $f(P) \leftarrow$ ***undefined***

This approach will create a zero set which is piecewise linear. It may contain discontinuities where $f(P)$ is undefined. However, according to Hoppe et.al. these discontinuities do not adversely affect the algorithm.

The last step in this algorithm consisting of contour tracing is described; however, no details are given. As a result, we will do the same. We define cubes throughout our modified point cloud.

These cubes are defined with edge length no larger than $\varepsilon + \delta$. The algorithm visits cubes that intersect $Z(f)$ by pushing onto a queue only appropriate neighboring cubes. As each neighbor is visited, a new edge is created by connecting the previous element of $Z(f)$ to the neighboring element. For example, suppose $z_i \in Z(f)$ and $z_j \in Z(f)$ are two distinct elements in the zero set of f. Furthermore, suppose C_i and C_j are cubes intersecting z_i and z_j respectively. If C_i and C_j are neighbors, an edge $e_{i,j}$ will be created. Otherwise, no edge will be created: Finally, the edges are collapsed, the process of which is left undiscussed, and the resulting triangulation is our final reconstruction. Notice that the final reconstruction lacks the sharp edges of the object being reconstructed. This is a consequence of the edge collapse. Thus, we may not always want to collapse the edges as was done in this case. See figures 14, 15 and 16 from [HOP92].

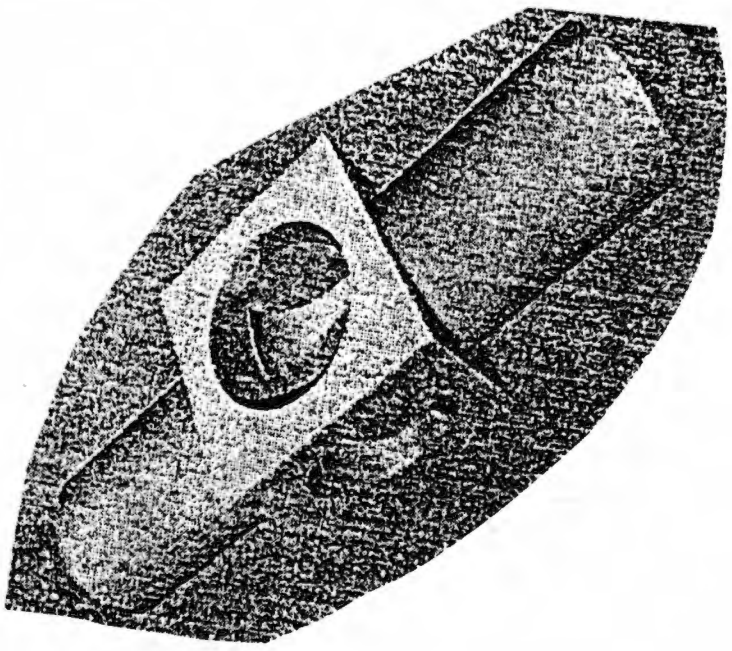

Figure 14: Original Object

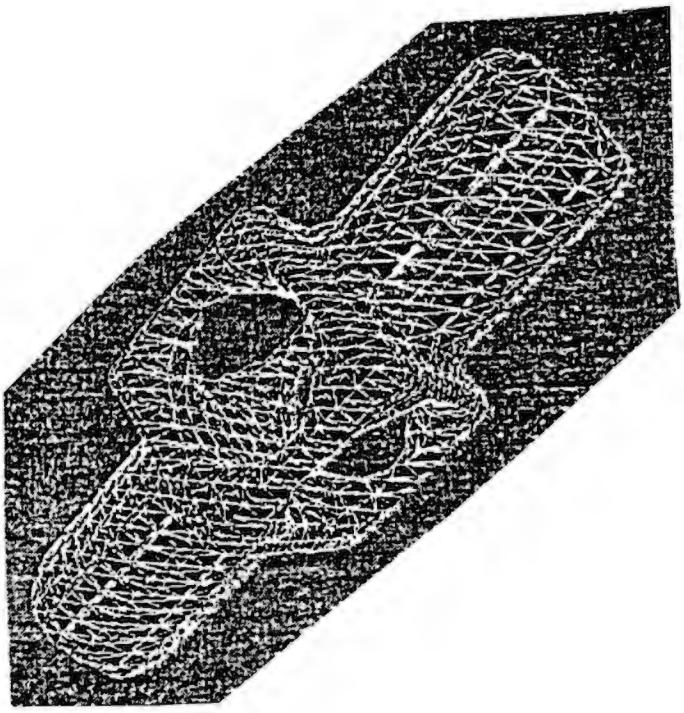

Figure 15: Output of Cube March

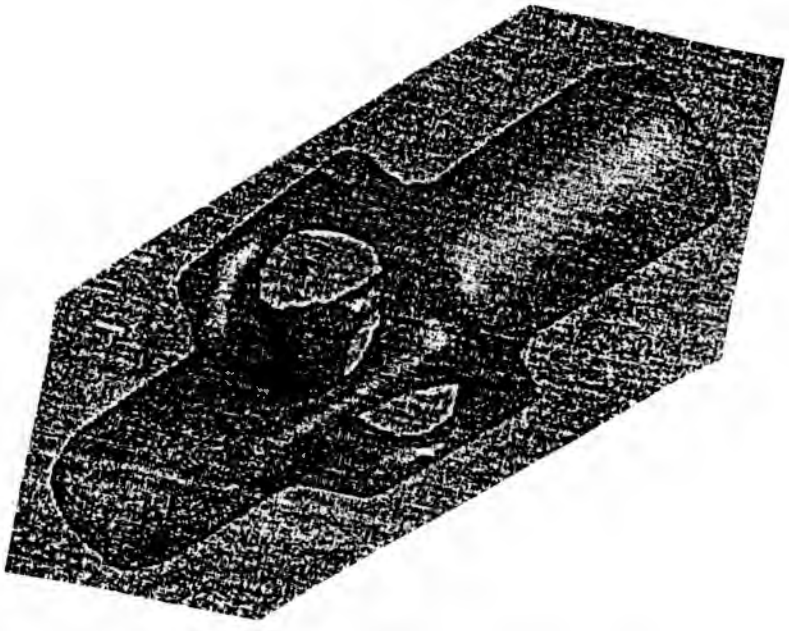

Figure 16: Final Surface after Edge Collapse

5.2.2 Method 2

The approach developed by Gopi et.al. [GP00] will be presented in two phases. First, we will consider the sampling criteria developed to accommodate changes in the surface Σ which is supposed to ensure a more accurate recreation. After that, we will proceed with the algorithm used to recreate the surface. Before we begin, we want to specify the assumptions about the surface we are recreating and give a brief overview of the algorithm. First, we will state our assumptions about the surface and data set:

1. We assume proper sampling. This is determined by the sample density δ defined at the beginning of this section.

2. We assume that Σ is smooth and the ratio of the maximum to minimum principal curvature at any point is bounded by some constant α.

3. We assume that for small δ, an arc from p to p_i on Σ can be replaced by an edge p to p_i.

The surface recreation algorithm takes as input a point cloud gathered by a range camera. The output is a set of triangles passing through our data which will represent our surface. The algorithm proceeds through four steps:

1. Normal Computation: This was discussed in section 3.1.2

2. Candidate Point Selection: This step will choose points which may be possible neighbors to a vertex in the final triangulation Using the sample criteria, described later, we will compute the candidate point set for every point $p \in PC$.

3. Neighbor Computation: We will map each point in the point set onto the tangent plane to compute the neighbors. We will do this for each point in the point cloud.

4. The final triangulation will be determined from the neighbor relationships found in step 3.

Now that we have an overview of the algorithm, we will begin with a discussion about the density δ of our point cloud. The sampling criteria is based on the normal curvature.

Sampling Criteria Because we will be working in the tangent plane, we want to ensure proper sampling in areas of high normal curvature. Normal curvature describes how fast the surface "bends" away from the tangent plane. Essentially, the larger the value of the normal curvature, either positive or negative, the faster the surface pulls away from the tangent plane. As a result, in areas of high curvature we want the data points to be closer together to accommodate this bending. This will ensure a more accurate reproduction of our surface without "flattening out" these areas of high curvature. Accommodating these areas of high curvature is an essential element of this method. The proofs in this section are mathematically unsatisfying. However, they are included because portions of them appear in [GP00]. This author simply completes the portions not specified but makes no assertions about their viability.

The foundation of the sampling criteria is the normal curvature of the surface being sampled. To fully utilize the results, the normal curvature of the surface in the areas of highest curvature should be estimated before the sampling from the range camera can begin. However, this is usually not possible in general surfaces being reconstructed. As a result, in practice, most of the following discussion must be applied to an initial point cloud.

This discussion will take place in the context of classical differential geometry. As such, we will assume smoothness. To make the assertions simpler to show, we first will show that we can express the surface in local neighborhoods as a height function in terms of the normal curvature k_{n_p}.

Lemma 73 *A surface Σ can be approximated locally as a height function $h(r,\theta) \approx \frac{k_{n_p}(\theta)r^2}{2}$ where $k_{n_p}(\theta)$ is the normal curvature at p in the direction θ.*

Proof. Let Σ represent our original surface and let $p \in \Sigma$. WLOG, assume $p = (0,0,0)$ is the origin, the tangent plane $\mathbf{T}(p)$ is the $z = 0$ plane, and the two principal directions of Σ at p are $\mathbf{e}_{p_1} = (1,0,0)$ and $\mathbf{e}_{p_2} = (0,1,0)$. From DoCarmo [DC76], we know that there is a small neighborhood centered at p, denoted by W_p, such that the projection map $\pi : (x,y,z) \to (x,y)$ is one-to-one and a diffeomorphism with open image $V_p \subset \mathbb{R}^2$.

Let the point $(x,y,z) \in \Sigma$ be a point in the local neighborhood of p defined in a local coordinate system at p. Since π is a diffeomorphism, π^{-1} exists and is smooth. Since π^{-1} is open in $\mathbb{R}^3$, is differentiable and $\pi^{-1}(p) = (0,0,0)$, by the implicit function theorem, there exists a function

$h : (x, y) \to \mathbb{R}$ such that we can approximate the surface in the neighborhood of p by

$$W_p(x, y) = \{(x, y, h(x, y)) \forall (x, y) \in V_p\}$$

The tangent plane is given by the basis vectors

$$\mathbf{T}_{1_W} = \left(1, 0, \frac{\partial h}{\partial x}(0, 0)\right)$$
$$\mathbf{T}_{2_W} = \left(0, 1, \frac{\partial h}{\partial y}(0, 0)\right)$$

Because we assumed that the tangent plane at p is the $z = 0$ plane, $\frac{\partial h}{\partial x}(0, 0) = \frac{\partial h}{\partial y}(0, 0) = 0$.

The shape operator (see [ON83]) applied to our basis vectors is defined as

$$S_p \mathbf{e}_{p_1} = \frac{\partial^2 h}{\partial x^2}(p)\mathbf{e}_{p_1} + \frac{\partial^2 h}{\partial x \partial y}(p)\mathbf{e}_{p_2}$$
$$S_p \mathbf{e}_{p_2} = \frac{\partial^2 h}{\partial x \partial y}(p)\mathbf{e}_{p_1} + \frac{\partial^2 h}{\partial y^2}(p)\mathbf{e}_{p_2}$$

Because $\mathbf{e}_{p_1}$ and $\mathbf{e}_{p_2}$ are the principal directions, we know $\frac{\partial^2 h}{\partial x \partial y} = 0$, $\frac{\partial^2 h}{\partial x^2}(0, 0)$, and $\frac{\partial^2 h}{\partial y^2}(0, 0)$ are the principal curvatures, denoted by k_{p_1} and k_{p_2} respectively.

Using Taylors formula, we expand $h(x, y)$ about the origin $(0, 0)$

$$h(x, y) = h(0, 0) + x\frac{\partial h}{\partial x}(0, 0) + y\frac{\partial h}{\partial y}(0, 0) + \frac{1}{2}\left(x^2\frac{\partial^2 h}{\partial x^2} + 2xy\frac{\partial^2 h}{\partial y \partial y} + y^2\frac{\partial^2 h}{\partial y^2}\right)$$
$$+ \text{ Higher order terms}$$
$$= \frac{1}{2}\left(k_{p_1}x^2 + k_{p_2}y^2\right) + \text{ Higher order terms}$$

If we convert to polar coordinates where $r = \sqrt{x^2 + y^2}$ and θ is the angle the vector determined by point (x, y) makes with the x-axis, or $\mathbf{e}_{p_1}$, we obtain

$$h(r, \theta) = \frac{1}{2}\left(k_{p_1}r^2\cos^2\theta + k_{p_2}r^2\sin^2\theta\right) + \text{ Higher order terms}$$
$$\approx \frac{r^2}{2}\left(k_{p_1}\cos^2\theta + k_{p_2}\sin^2\theta\right)$$

In section 2.2, we found that the Euler Equation relates normal curvature k_{n_p} to the direction $\mathbf{v}$, here $\mathbf{v}$ is the vector determined by (x, y). The Euler Equation is given by

$$k_{n_p}(\theta) = k_{p_1}\cos^2\theta + k_{p_2}\sin^2\theta$$

Thus,

$$h(r,\theta) \approx \frac{k_{n_p}(\theta)\, r^2}{2}$$

This completes the proof. ■

The ability to represent the surface as a local height function based on normal curvature allows us to set sampling criteria more easily. We would like to be able to describe the behavior of the normal vector, $\mathbf{N}_p$, along a space curve α on surface Σ through p. [MIL77] tells us that we can describe this behavior by the derivative of the Frenet-Serret apparatus,

$$\mathbf{N}'_p(\alpha) = -k_{n_v}\partial\alpha\overrightarrow{T} - t\partial\alpha\overrightarrow{B} \text{ or}$$
$$\left|\mathbf{N}'_p(\alpha)\right| = \sqrt{k_{n_v}^2 + t^2}\,|\partial\alpha|$$

Where t is the geodesic torsion and $|\partial\alpha|$ is the arclength. By definition (see [MIL77]), $\sqrt{k_{n_v}^2 + t^2}$ is called the total curvature of α. In our case, we can simplify the equation for planar curves through p where $t = 0$, so we obtain

$$\begin{aligned}\left|\mathbf{N}'_p(\alpha)\right| &= \sqrt{k_{n_v}^2}\,|\partial\alpha| \\ &= k_{n_p}\,|\partial\alpha|\end{aligned}$$

We assume $\left|\mathbf{N}'_p(\alpha)\right|$ is some constant at p, so replace $|\partial\mathbf{N}_p(\alpha)|$ by δ. We also assume p is a regular point; as such, the neighborhood of p is homeomorphic to an open disk. Let $B_\delta(p)$ be a closed ball centered at p with radius δ. We will call $B_\delta(p)$ the *immediate δ-neighborhood* of p Let $q \in B_\delta(p)$. We will replace the arclength $|\partial\alpha|$ by the Euclidean arc length $|pq|$ to obtain

$$\delta = k_{n_p}\,|pq| \,\forall q \in B_\delta(p)$$

Definition 74 *We will call a point cloud a* **δ-sample** *if every point $p \in \Sigma$ has a closest point q in the point cloud such that $q \in B_\delta(p)$. δ can be changed to obtain different sampling densities of the surface.*

We will use the following proposition to find an initial estimate for δ.

Proposition 75 *Suppose $p = (0,0,0)$ is the origin and the normal vector at p is $\mathbf{N}_p = (0,0,1)$ and $q = (x, y, h(x,y))$, where $q \in B_\delta(p)$. Then $\delta \approx k_{n_p}\sqrt{r^2 + \frac{k_{n_p}^2 r^4}{4}}$.*

Proof. Let $p = (0,0,0)$, $\mathbf{N}_p = (0,0,1)$, and $q = (x, y, h(x,y))$ where $q \in B_\delta(p)$. We know $h(r,\theta) \approx \frac{k_{n_p}(\theta) r^2}{2}$ where k_{n_p} is the normal curvature at p in the direction of θ and $r = \sqrt{x^2+y^2}$. Then

$$\begin{aligned}
\delta &= k_{n_p}(\theta)\,|pq| \\
&= k_{n_p}(\theta)\sqrt{q \cdot q} \text{ since } p = (0,0,0) \\
&= k_{n_p}(\theta)\sqrt{(x,y,h(r,\theta)) \cdot (x,y,h(r,\theta))} \\
&= k_{n_p}(\theta)\sqrt{(x^2+y^2+h^2(r,\theta))} \\
&\approx k_{n_p}(\theta)\sqrt{\left(r^2 + \frac{k_{n_p}^2 r^4}{4}\right)}
\end{aligned}$$

■

Now we will discuss some properties of a δ-sample.

Proposition 76 *Consider two points $q \in B_\delta(p)$ and $r \in B_\delta(p)$. Then the maximum ratio of edge distances $\frac{|pq|}{|pr|}$ is bounded above by a function of the principal curvatures. Specifically, $\frac{|pq|}{|pr|} \leq \frac{k_{\max q}}{k_{\min r}}$.*

Proof. Because we assumed $\delta = k_{n_p}|pq|\ \forall q \in B_\delta(p)$ is constant, we have $k_{n_q}|pq| = k_{n_r}|pr| = \delta \Rightarrow \frac{|pq|}{|pr|} = \frac{k_{n_q}}{k_{n_r}}$, where k_{n_q} and k_{n_r} are the curvatures at q and r in the direction of p. It is easy to see, by definition, that this ratio will reach a maximum when k_{n_q} is the maximum principal curvature and k_{n_r} is the minimum principal curvature at p. Thus, the maximum ratio at every point is the ratio of the principal curvatures at that point. ■

Proposition 77 *Let $\mathbf{N}_p$ be the unit normal to Σ at p. Let $\overrightarrow{pq}$ be the vector from p to q such that $q \in B_\delta(p)$. Define the height function to be the function $H_p(q) = |\mathbf{N}_p \cdot \overrightarrow{pq}|$. Then, $H_p(q) \leq \frac{\sqrt{1+\delta^2}-1}{k_{\min_p}}$.*

Proof. Assume, WLOG, that $p = (0,0,0)$ is the origin and the normal vector at p is $\mathbf{N}_p = (0,0,1)$ and $q = (x, y, h(x,y))$. This can be done for any $p \in \Sigma$ and $\mathbf{N}_p$ by a simple translation and rotation. Then $H_p(q) = h(x,y)$ since p is the origin and $\mathbf{N}_p = (0,0,1)$. We shown that shown

that $\delta \approx k_{n_p}\sqrt{r^2 + \frac{k_{n_p}^2 r^4}{4}}$. Thus

$$\begin{aligned}
\delta^2 &\approx k_{n_p}^2\left(r^2 + \frac{k_{n_p}^2 r^4}{4}\right) \Rightarrow \\
\delta^2 &\approx k_{n_p}^2 r^2 + \frac{k_{n_p}^4 r^4}{4} \\
&= 2k_{n_p}h + k_{n_p}^2 h^2 \Rightarrow \\
0 &\approx k_{n_p}^2 h^2 + 2k_{n_p}h - \delta^2
\end{aligned}$$

The positive solution is

$$\begin{aligned}
h &= \frac{-2k_{n_p} + \sqrt{4k_{n_p}^2 + 4k_{n_p}^2\delta^2}}{2k_{n_p}^2} \\
&= \frac{-2k_{n_p} + 2k_{n_p}\sqrt{1+\delta^2}}{2k_{n_p}^2} \\
&= \frac{\sqrt{1+\delta^2} - 1}{k_{n_p}}
\end{aligned}$$

It reaches a maximum when the normal curvature is minimum, which is $k_{\min_p}$. Thus, h is bounded by

$$\frac{\sqrt{1+\delta^2} - 1}{k_{\min_p}}$$

■

This result can be used to define how close two different parts of the surface can come so a δ-sampling is sufficient for our reconstruction algorithm. Since the maximum height value for any point in $B_\delta(p)$ is bounded by $h = \frac{\sqrt{1+\delta^2}-1}{k_{\min_p}}$, we bound the distance between the two samples of the parts to be greater than $2h$.

Proposition 78 *Define the angle function $D(p,q) = |\mathbf{N}_p \cdot \widehat{pq}|$, where $\widehat{pq}$ is the* ***unit*** *vector from p to q. Then $D(p,q) \approx \left(\frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1}\right)^{\frac{1}{2}} \approx \frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1}$ for all $q \in C_p^\delta$.*

Proof. As before, assume p is the origin, $p = (0,0,0)$, $\mathbf{N}_p = (0,0,1)$ and $q = (x,y,h(x,y))$. Then as we have just seen, $D(p,q) = |\mathbf{N}_p \cdot \widehat{pq}| = \frac{h(x,y)}{\sqrt{x^2+y^2+h^2(x,y)}}$, (remember that $\widehat{pq}$ is the **unit**

vector). Then, as we have done in the previous proof, we obtain

$$\begin{aligned} D^2(p,q) &= \frac{h^2(x,y)}{x^2+y^2+h^2(x,y)} \\ &\approx \frac{1}{r^2+\frac{k_n^2 r^4}{4}}\left(\frac{k_n^2 r^4}{4}\right) \\ &\approx \frac{k_n^2 r^2}{4+k_n^2 r^2} \end{aligned}$$

Substituting $k_n^2 r^2$ by d, we obtain

$$\begin{aligned} D^2(p,q) &= \frac{d}{4+d} \\ D(p,q) &= \frac{d}{\sqrt{4d+d^2}} \end{aligned}$$

We know from the previous proof that

$$\begin{aligned} \delta &= k_{n_p}\sqrt{r^2+\frac{k_{n_p}^2 r^4}{4}} \\ \delta^2 &= k_n^2 r^2 + \frac{k_{n_p}^4 r^4}{4} \\ &= d + \frac{d^2}{4} \\ 0 &= d^2 + 4d - 4\delta^2 \end{aligned}$$

The positive root of this equation is the constant

$$\begin{aligned} d &= \frac{-4+\sqrt{16+16\delta^2}}{2} \\ &= -2+2\sqrt{1+\delta^2} \\ &= 2\left(\sqrt{1+\delta^2}-1\right) \end{aligned}$$

Now we plug this value into the original $D^2(p,q)$

$$\begin{aligned} D^2(p,q) &= \frac{h^2(x,y)}{x^2+y^2+h^2(x,y)} \\ &\approx \frac{d}{4+d} \\ &= \frac{2\left(\sqrt{1+\delta^2}-1\right)}{4+2\sqrt{1+\delta^2}-2} \\ &= \frac{2\left(\sqrt{1+\delta^2}-1\right)}{2\left(\sqrt{1+\delta^2}+1\right)} \\ &= \frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1} \\ D(p,q) &\approx \left(\frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1}\right)^{\frac{1}{2}} \\ &\approx \frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1} \end{aligned}$$

This completes the proof ■

This last proof shows us that all the points in $B_\delta(p)$ make the same angle with the normal vector at p. This makes sense because they lie in the plane normal to $\mathbf{N}_p$, i.e. in the same plane as p. Recall that in our development of δ, we assumed the curve s to be a planar curve containing p. The normal vector to p, $\mathbf{N}_p$, is unique; thus for every $q \in B_\delta(p)$, there is some curve s containing p and q that is planar, and, therefore, in the plane normal to $\mathbf{N}_p$.

This proposition will allow us to further define our value for δ. If we know two different parts of a surface are within distance d from each other, we impose a δ-sampling $\delta \leq \frac{\sqrt{d^2 k_{\min_p}^2 + 4dk_{\min_p}}}{2}$.

Corollary 79 *If two parts of a surface are within distance d from each other, we set*

$$\delta \leq \frac{\sqrt{d^2 k_{\min_p}^2 + 4dk_{\min_p}}}{2}$$

Proof. We want to set the smallest value for δ. We know two parts of the surface are within

distance $d = k_n^2 r^2$. In this instance,

$$\begin{aligned}\delta &= k_{n_p}\sqrt{r^2 + \frac{k_{n_p}^2 r^4}{4}} \\ &= k_{n_p}\sqrt{4r^2 + k_{n_p}^2 r^4} \\ &= \sqrt{4d + d^2} \\ &\leq \frac{\sqrt{d^2 k_{\min_p}^2 + 4dk_{\min_p}}}{2}\end{aligned}$$

■

Remark 80 *Taken in totality, the propositions show that the points in the neighborhood of p on a δ-sampled surface satisfy strict bounds on the discretized normal curvature at p. Notice that in each of the above propositions, k_{n_p} is taken to be some number. However, in the continuous case, it is a function of direction, so it is not independent of direction θ, so in the propositions, we had to alter them to specify a specific direction. If we are to accept them as stated in [GP00], the propositions will be valid only if we discretize k_{n_p}. Because we must calculate the curvature values on our discrete set of data, which we will discuss shortly, k_{n_p}, $k_{\max_p}$ and $k_{\min_p}$ are naturally discrete.*

Furthermore, these propositions also force us to assume that the local neighborhood of p is planar. This is generally not true for regular surfaces in $\mathbb{R}^3$. For the purposes of recreating a surface based on discrete data, this assumption will not affect the end result because the triangulation consists of a set of planar triangles.

The entire discussion in this section would be much more relevant if there were methods by which to estimate curvature values before taking a δ-sample. However, doing this is not possible because we do not have a parametrization of the surfaces we would be recreating, nor do we have methods to measure the curvature on the surface itself. Thus, we are forced to use data points from the point cloud to make our initial estimates. As such, the value we obtained

$$\delta \approx k_{n_p}\sqrt{r^2 + \frac{k_{n_p}^2 r^4}{4}}$$

may be very difficult to achieve in practice.

These issues are not discussed in the paper [GP00].

If various portions of the surface fail to meet the sampling criteria, further samples of those portions of the surface can be taken. It is questionable as to whether or not using our point cloud will make any difference on our sampling density. Furthermore, obtaining a point cloud, and then using the same point cloud to determine whether or not it is dense enough in particular areas is mathematically unappealing. Nevertheless, with current technology, it is our only option. The concerns about obtaining a sufficient sample of the surface in high normal curvature areas are well-founded because the neighborhoods in these areas pull away from the tangent plane more rapidly. Therefore, we would want our data to be more dense in these areas. One method by which this could be accomplished is by visually locating these high curvature areas and setting the range camera to increase the density of our data in these areas.

The Triangulation We will now begin discussing the algorithm of the triangulation process. We will assume a proper δ-sample throughout this discussion. Our first task is to estimate the unit normal vector, $\mathbf{N}_p$, at each $p \in PC$. We propagate the normal vectors to obtain a global orientation using the same method as Hoppe et.al. [HOP92] discussed in the previous section. Thus, we also estimate the tangent plane, $\mathbf{T}(p)$ at each $p \in PC$, also using the same method as Hoppe et.al.. Finally, we estimate the principal curvatures at each point.

We now have at each point $p \in PC$ an associated normal vector, $\mathbf{N}_p$, and an associated tangent plane, $\mathbf{T}(p)$. We also have at each $p \in PC$ estimates of principal curvatures, $k_{\min_p}$ & $k_{\max_p}$, and principal directions, $\mathbf{e}_{\min_p}$ & $\mathbf{e}_{\max_p}$. Given a δ-sample of a surface, we consider all points within a 2δ neighborhood of $p \in PC$ as possible neighbors of p. By this, we mean that we will consider all points p_j, $j = 1 \ldots n$, such that $d(p_j, p) < 2\delta$.

We have shown that the maximum ratio of distances between two points in $B_\delta(p)$ is bounded by $\frac{k_{\max_p}}{k_{\min_p}}$. Since we are considering all points within a 2δ neighborhood, the distances are bounded by $d_p = \frac{2k_{\max_p}}{k_{\min_p}}$. We will prune this set of points so we can choose neighbors of p more efficiently, and we will use d_p in this process.

Let s denote the distance from p to its nearest neighbor. The pruning stage begins by defining a box centered at p with the lengths of the sides $L = 2d_p s$ and returning all the data points inside it. The next pruning stage defines a sphere with radius $d_p s$ and rejects all points outside it. This

sphere is called the *sphere of influence,* and it is contained inside our box:

Definition 81 *The* ***sphere of influence*** *contains all points less than distance $d_p s$ from p, where $d_p = \frac{2k_{\max_p}}{k_{\min_p}}$ and s is the distance from p to its nearest neighbor.*

We prune the set further by computing the height values of these points in the local tangent plane of p. If the height value is greater than $\frac{\sqrt{1+\delta^2}-1}{k_{\min_p}}$ (developed in the previous section), they are removed from consideration as possible neighbors of p. The points that remain are possible neighbors of p.

We are left with a small set of candidate neighbors of p and are ready for the triangulation process. These candidate points are transformed into a local coordinate system with p at the origin and mapped to the tangent plane at p using the same method as Hoppe [HOP92]. These points now have 2 dimensional coordinates. The projected point set is partitioned on the basis of which quadrant they lie in with respect to this local coordinate system. We create a triangulation method by ordering the points according to angle. Using the square of the sine function, we discretize the angles between 0 and $\frac{\pi}{2}$ and store the values in a table. We use the table to order the points in each quadrant by angle. If a value is between two values in the table, we approximate the angle linearly. This ordering determines how we proceed testing each point to see if it is a neighbor of p and should be a vertex of a triangle at p.

Now that we have ordered the points in each quadrant, we can proceed with the triangulation. Given three consecutive points in angle order p_{i-1}, p_i, p_{i+1}, we check to see whether the middle point, p_i, is a neighbor to p. The algorithm is as follows:

Algorithm 82 *Check Neighbor:*

$L_{p_{i-1}}$ = *Perpendicular Bisector of line segment* pp_{i-1}

L_p = *Perpendicular Bisector of line segment* pp_i

L_{p+1} = *Perpendicular Bisector of line segment* pp_{p+1}

$I_{p_{i-1},p_{i+1}}$ = *Intersection point of* $L_{p_{i-1}}$ *and* $L_{p_{i+1}}$

L_{Ip} = *Line parallel to* L_p *and passing through* $I_{p_{i-1}p_{i+1}}$

M_p = *Midpoint of the line segment* pp_i

If *both p and* M_p *lie on the same side of* L_{Ip}

Then *p is a neighbor to p when compared with* p_{i-1} *and* p_{i+1}

Else *p is not a neighbor to p*

This algorithm is explained pictorially in figure 17 adapted from [GP00]. If $\{p_1, \ldots, p_n\}$ are the ordered projected candidate neighbors of p, then the above algorithm is performed for every triplet p_{i-1}, p_i, p_{i+1}. If p_i is calculated to be a neighbor of p, then the triplet p_i, p_{i+1}, p_{i+2} is tested. If p_i is not calculated to be a neighbor of p, then p_i is rejected and we test $p_{i-2}, p_{i-1}, p_{i+1}$ to test p_{i-1} for viability as a neighbor of p. The neighbors of each vertex is stored in a table ordered by angle with p_{i-1}, p_i, p_{i+1} forming a triangle. Once this is done for each point in the point cloud, the surface reconstruction is complete.

The most difficult part of this algorithm is from the beginning when setting the sampling criteria. As we previously stated, using our original point cloud to determine whether or not it is sufficient may not make much of a difference. We could simply choose the areas of high curvature before taking our original sample and program our equipment to obtain higher sampling density in these areas. We can choose these areas visually. The properties of a δ-sample are also used in the algorithm in determining the neighbors of each $p \in PC$. Using these properties is an effective way to prune our data set. For an example of the result of this method, refer to figure 18.

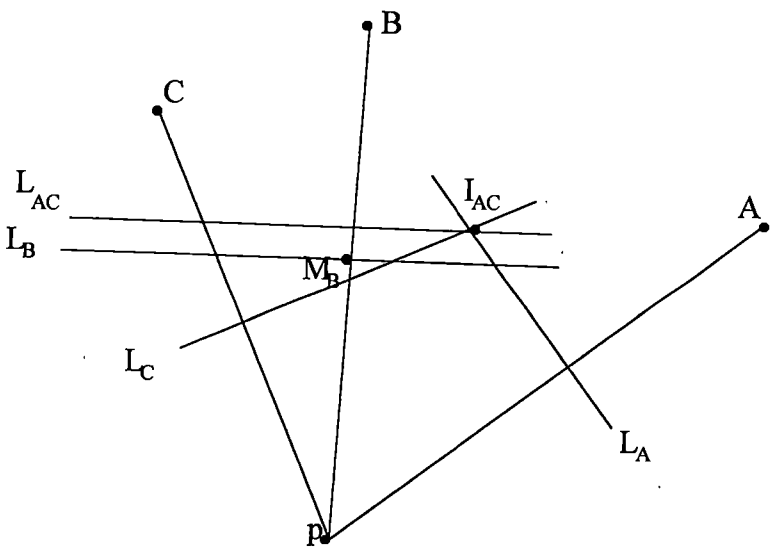

Figure 17: Finding Neighbors of p. Here $B = p_i, A = p_{i-1}, C = p_{i+1}$

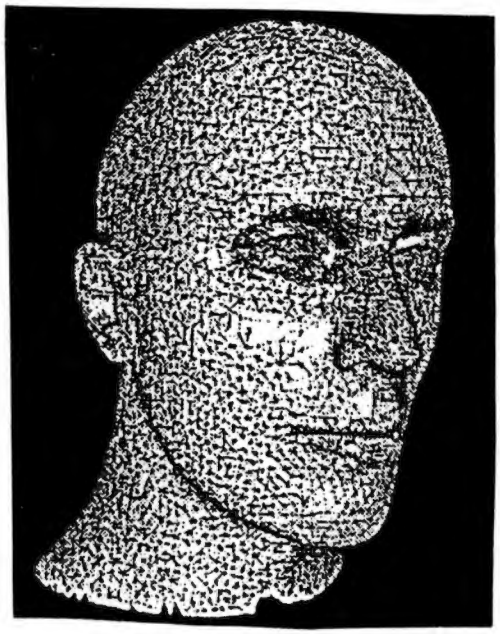

Figure 18: A Triangulation of a Surface

5.2.3 Method 3

Our final triangulation method simulates a sphere rotating on the point cloud to create the triangulation.

Remark 83 *We will refer to this sphere as a* ***ball*** *to remain consistent with the name of the algorithm.*

The Ball Pivoting Algorithm (BPA) was designed by engineers who were commissioned to create a computer model of Michelangelo's *Florentine Pieta.* It was created by Bernardini et.al [BN99]. We begin with a point cloud which is δ-dense, so that a ball with radius δ, denoted by B_δ, can rotate on the points without falling through. Recall that a point cloud is δ-dense if $d(p, PC) \leq \delta$ for $p \in PC$. In this case, the value for δ is determined by the user.

To accommodate uneven sampling, the algorithm can be run again with B_ω where $\omega \neq \delta$. We calculate normal vectors at points $p \in PC$ and orient them in the same direction to ensure an oriented recreation. This method does not develop a specific algorithm to calculate the normal

vectors. As such, any reliable method is sufficient. As the ball passes over the point cloud, it connects the points it comes into contact with to create the triangulation. This algorithm makes no estimation of curvature. The normal vectors are used to ensure that the initial triangle is consistently oriented with its three vertices and to determine which triangles to reject in the final triangulation. A triangle is rejected if the dot product with the triangle normal, $\mathbf{N}_{\mathbf{T}_p}$, and vertice normals is negative.

Let $PC = \{p_1, p_2, \ldots, p_n\}$ and $\{\mathbf{N}_{p_1}, \mathbf{N}_{p_2}, \ldots, \mathbf{N}_{p_n}\}$ be their associated normal vectors. The first challenge is determining a seed triangle

Definition 84 *We call a* ***seed triangle*** *our initial triangle.*

For each connected component of the surface, we locate one seed triangle. From this initial seed triangle, we will create the triangle mesh for the entire connected component. We begin by selecting an arbitrary $p \in PC$ not yet used in the triangulation. Now consider all pairs of points $p_i, p_j \in nbhd(p)$ in order of distance from p and build a potential triangle with vertices $\{p, p_i, p_j\}$. This will be our seed triangle if the $\{\mathbf{N}_p, \mathbf{N}_{p_i}, \mathbf{N}_{p_j}\}$ and $\mathbf{N}_{\mathbf{T}_p}$ are consistently oriented. Finally, we test to make sure the ball touches $\{p, p_i, p_j\}$ but not any other $p_k \in PC$. If one of these conditions is not satisfied, we test another seed triangle. Once a seed triangle is found, store the edges and proceed. These edges are stored in a linked list called the *Front.*

Definition 85 *We call the* ***Front****, denoted by $\mathcal{F}$, a collection of linked lists of edges, their opposite vertex, the center of the ball that touches all three points, and links to the previoius and next edge. If p_i and p_j are an edge of a triangle, it is stored in $\mathcal{F}$ as (p_i, p_j). We will denote the edge, $e_{(i,j)}$.*

Now that we have our seed triangle, we can proceed with the triangulation. We began with a ball of radius δ resting on all three vertices of a triangle. Assume $e_{(i,j)}$ is the pivoting edge. Staying in contact with the two endpoints p_i and p_j, we pivot B_δ until it comes into contact with another point, call it p_k. This will form a new triangle with vertices p_i, p_j, p_k. If no point is hit, the edge $e_{(i,j)}$ is labeled a boundary edge.

Definition 86 *We call an edge* ***active*** *if it is used for pivoting.*

Definition 87 *We call an edge* ***boundary*** *if it is not possible to pivot from it.*

Of course we cannot roll a ball on our point cloud, so we want to develop a way to simulate this process. First, we will calculate the midpoint, m_{ij}, of $e_{(i,j)}$

$$m_{i,j} = \frac{1}{2}(p_j + p_i)$$

Next, we consider all points within a 2δ neighborhood of m_{ij}. For each such point p_x, compute the center c_x of the ball touching p_i, p_j, p_x. Each c_x should lie on the circular trajectory, γ, around $m_{i,j}$ and can be computed by intersecting a δ-ball centered at p_x with circular trajectory γ. Of these centers, c_x, we pick the one that is first along the trajectory, and choose the first point p_k that is hit. For a pictorial explanation, see the figure 19 adapted from [BN99].

We perform two operations which affect $\mathcal{F}$. They are called *join* and *glue*. The easier one is *join*. When the ball pivots around edge $e_{(i,,j)}$ and touches a new *unused* vertex p_k. In this case, we have developed a new triangle with $\{p_i, p_j, p_k\}$ as the vertices. We update $\mathcal{F}$ by removing edge $e_{(i,,j)}$ and adding the two edges $e_{(i,k)}$ and $e_{(k,j)}$. For an example, see figure 20 adapted from [BN99].

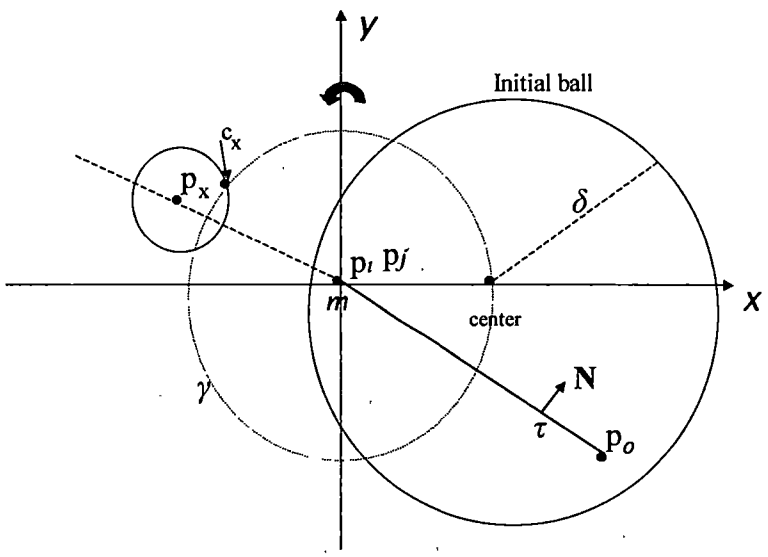

Figure19: The Ball Pivoting Operation

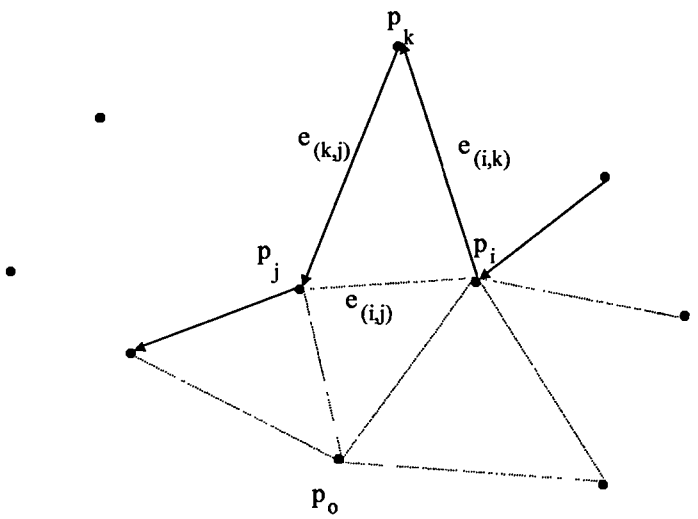

Figure 20: The Join Operation

When p_k is already part of the mesh, one of two cases arises:

1. p_k is an internal mesh vertex, i.e. no front edge uses p_k. This means that it is only an oppostie point and is not used as a a pivot edge vertex. The corresponding triangle cannot be created. In this case $e_{(i,j)}$ is marked as a boundary edge.

2. p_k belongs to the front. In other words, it is used as a vertex on a pivoting edge. After checking edge orientation with the corresponding vertices, we apply a *join* operation and output the new mesh triangle with vertices p_i, p_j, p_k. This could potentially create pairs of coincident edges with opposite orientation which are removed by the glue operation. Coincident edges are edges $e_{(i,k)}$ and $e_{(k,i)}$.

As stated in the above case, *glue* removes from the front pairs of coincident edges with opposite orientation. For example, when edge $e_{(i,k)}$ is added to $\mathcal{F}$ by a *join* operation. If edge $e_{(k,i)}$ is on $\mathcal{F}$, *glue* will remove the pair of edges $e_{(i,k)}$ and $e_{(k,i)}$ and update $\mathcal{F}$ accordingly. For an example of the glue operaton, see figure 21 adapted from [BN99].

The process is completed when all points in the point cloud have been visited and there are no coincident edges . In some cases, the δ-sample is uneven so the ball may not visit all data points on a connected component or may "fall through" the data set and we may have to select another seed triangle. This may cause the algorithm to create two separate components when there should only be one. When this happens and we have exhausted all points that could create triangles using B_δ, we can run the algorithm again with a larger ball.

This process seems to be intuitive and efficient. It allows the technology to do all the work and because no manipulation of the data is required. The only calculations are those to estimate the normal vectors and midpoints of line segments. Tangent planes, which are used in the previous two algorithms, can introduce errors because assumptions are needed about the data itself. No such assumptions are necessary with this algorithm. The algorithm does not attempt to compensate for noise, but it does assume that a preprocessing algorithm was performed to filter out the outliers (those points which are created that are a large distance from the point cloud). An example of the result of this process is shown in figure 22 from [BN99].

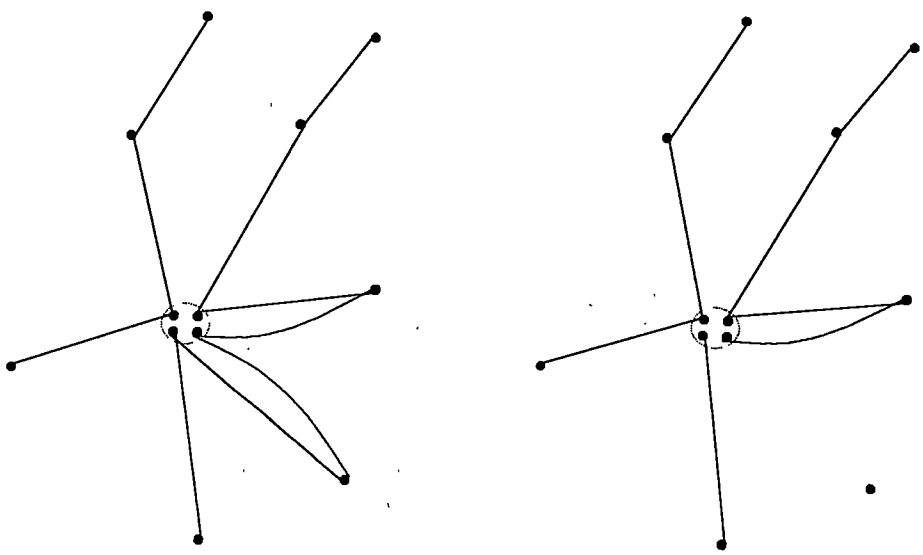

Figure 21: The Glue Operation. Points in the circle represent the same point

Figure 22: Output of the BPA

This concludes our discussion of triangulation methods.

5.3 Conclusion

Each of the methods described in this chapter was developed for a specific purpose. Each one works well for the specific purpose for which it was created. However, as we can readily see from this last approach, if we want a detailed, accurate model of the surface, this procedure would be highly insufficient. Similarly, if we need a speedy algorithm for a robot to recognize that objects are in its path, the first two methods discussed in this chapter require too much time. This lack of speed may cause the robot to run into the object before it is able to recognize how to avoid it or move it. The one algorithm which seems to satisfy both of these situations, and thus, almost any situation in general, is the BPA. It is efficient while producing a detailed reproduction. Specifically, in the robotic case where data is not attainable from every angle, we can make assumptions about the "unscanned" portions of the surface, such as those used in the last method about symmetry and continuity, to create a complete 3D reproduction of the surface so the robot can use the information

to either avoid the object or move it.

Overall, the function approximation method is less attractive and accurate than the others in this chapter. The function approximations are much more sensitive to noise, and thus, the values such as curvature and normal vectors are more subject to error. Furthermore, the strength of computers lie in their ability to handle large amounts of discrete data efficiently and effectively. The triangulation methods take full advantage of this strength. Errors can occur during the transition to the respective tangent planes and returning them in the end to their final positions. However, this is a small consideration because we can easily store these original positions.

6 CONCLUSION

6.1 Summary

This thesis has been an exploration into aspects of computerized 3-dimensional surface recreation. First,we discussed the methods by which data is collected for the computer to analyze. We found that, although not perfect, laser range cameras are preferred over the ultrasonic range cameras because of the resolution we are able to obtain. Furthermore, with advances in the technology, laser range cameras are becoming more accurate. We also discussed some of the terminology associated with these range cameras. After discussing the methods used to obtain the original data, we discussed some important definitions and theorems from differential geometry. The proofs of these theorems are not given because they can be found in any quality differential geometry text. With this foundation, we began discussing calculations on the data obtained from the range cameras.

In general, the first calculations in the reconstruction process are centered around estimating a normal vector at each element of the point cloud. Although differential geometry provides a direct method to calculate a normal vector from a differentiable parametrization of a surface, we were unable to utilize this tool because our data is comprised of discrete data points. We developed to two general categories. The first category estimates the normal vectors from this discrete set of points using difference equations or vector products [DEL94][HOP92][TB95]. The calculations were performed on the neighborhood of the point in question. The neighborhood was defined by either distance functions [HOP92][TB95] or by connectivity in the mesh [DEL94]. The second category estimates very small neighborhoods of the point in question by a differentiable parametrization and uses this parametrization to estimate the normal vector [FR93][FR98]. The approximating functions were found using least squares techniques which were not discussed in any detail.

Next, we discussed methods by which various curvature values are estimated. Again, two general categories were found. In this chapter, we began by discussing curvature estimation from approximating functions [FR93][FR98]. From the approximating functions, principal curvatures were estimated directly using the definitions from differential geometry. To reduce the influence of noise in our data, [FR98] attempts to refine the value of the principal curvature values by taking into consideration curvature values at another point in the data set. The second category approximated

curvature directly from a triangulation or simplex mesh [DEL94][TB95]. In this category, we began with a triangulation or simplex mesh before curvature was calculated. We estimated Gauss, Mean, and principal curvatures from this category. We discovered that of these two categories, the discrete case is probably more accurate.

Finally, we put it all together in Chapter 5 to recreate the surface. For each method, we found that the results from the triangulation techniques were close to the original surfaces being recreated. Since the goal of the approximating function was primarily to reproduce a rough image, this category was less accurate in reproducing the details of the suface, but was also sufficient for its purpose. The only technique we did not discuss until this chapter was the Ball Pivoting Algorithm. This technique was more of a combinatorial approach and relied very little on differential geometry. Nevertheless, it produced a triangulation very close to the very complicated surfaces being recreated. It was also more intuitive than the other algorithms. Although we attempted to address some of the questions, there are still many questions left unanswered.

6.2 Further Research

The more interesting approaches discussed in this thesis are the triangulation approaches which do not attempt to use approximating functions or volumetric primitives to recreate the surface. Although each method differs in the details, each one produces a reasonable approximation of the surface to be recreated. There may be some underlying facts about surfaces in general which support the discrete case. More study of differential geometry in this discrete case is warranted. This study could produce an approach to surface reconstruction which would produce accurate results for any general surface reconstruction application, reducing the need for different algorithms for each specific application.

Furthermore, many of the proofs in this thesis contain some assumptions about the surface which are generally untrue about surfaces in general. For example, normal curvature on a regular surface is not a constant at a particular point unless the normal curvature is discretized, although the assumptions in this case do not affect the outcome of the algorithm because normal curvature is discretized naturally by calculating curvature on the discrete set of data points. We may also want to explore the viability of determining whether a δ-sampled surface is dense enough by performing

calculations on that same δ-sample.

This author has learned that, in general, the proofs in all of these methods are mathematically incomplete and unsatisfying. Attempts have been made to complete these proofs, but in some cases this has been impossible because the arguments were difficult to follow.

By no means has this been an exhaustive study of all reconstruction algorithms. Other methods exist which would provide an interesting study. Two examples consist of methods discussed briefly in this thesis [DEL94] [FR98]. We have already given overviews of these methods and leave the details to the reader. Another method developed over several years would also provide an interesting study. Medioni et.al. [GUY97][MD00][TG96][TG99] have developed a tensor voting technique which seems to provide some promising results. Unlike the approaches discussed in this thesis, this one does not attempt to quantify curvature measurements. It categorizes each tensor, a symmetric covariance matrix, as either a stick, plate, or ball tensor at each element in the point cloud. These tensors are categorized according to their eigenvectors and eigenvalues. A stick tensor indicates that the element lies on a surface; a plate tensor indicates that the element lies on a curve, or at the intersection of two surfaces; and a ball tensor indicates that the element lies at the intersection of two curves.

The center of this process involves a voting procedure in which each element influences and refines the information associated with their neighbors. After the voting procedure, if the tensor is a stick, this means that the element associated with this tensor has a well defined normal vector associated with it, as does its neighbors. The confidence, or saliency, of the normal vector measurement is encoded in the tensor's eigenvalues. If the tensor is determined to be a plate, the element has a well defined tangent vector associated with it. Again, the confidence of this tangent direction is encoded in the tensor's eigenvalues. Finally, a ball tensor has no associated normal or tangent direction associated with it. These ball tensors help locate outliers and noise, and are generally ignored in the voting and reconstruction process. The mathematics involved with this method are very interesting. Unfortunately, this author did not have the time to do a thorough review of them to include a proper discussion of them in this thesis. However, the results seem to be promising and this method should be studied further.

References

References

[BN99] F. Bernardi, J. Mittleman, H. Rushmeier, C. Silva and G. Taubin. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization and computer Graphics,* 1999.

[BM95] A. Bluman. *Elementary Statistics, A Step by Step Approach.* William C. Brown Communications Inc., 1995.

[CS99] B. Chase. Calibration of Scanning Laser Range Cameras with Applications for Machine Vision. Master's Thesis, University of Tennessee, Knoxville, 1999.

[CL96] B. Curless and M. Levoy. A volumetric Method for Building Complex Models from Range Images. *Proceedings of ACM Siggraph,* Pages 393-312, 1996.

[DC76] M. DoCarmo. *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.

[DEL94] H. Delingette. Simplex Meshes: A General Representation for 3D Shape Reconstruction. *Inria Sophia Antipolis,* 1994.

[FLN89] P. Flynn and A. Jain. On Reliable Curvature Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Pages 110-116, 1989.

[FR93] F. P. Ferrie, J. Lagarde and P. Whaite. Darboux Frames, Snakes, and Super-Quadrics: Geometry From the Bottom Up. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Volume 15, August 1993.

[FR98] F. P. Ferrie, M. D. Levine. Deriving Coarse 3D Models of Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Pages 345-353, 1998.

[GB91] J. Goldberg. *Matrix Theory with Applications.* McGraw-Hill, 1991.

[GL83] G. Golub and C. VanLoan. *Matrix Computations.* Johns Hopkins Univ. Press, 1983.

[GP00] M. Gopi, S. Krishnan and C. T. Silva. Surface Reconstruction Based on Lower Dimensional Localized Delaunay Triangulation. *Eurographics 2000,* Volume 19, 2000.

[GUY97] G. Guy and G. Medioni. Inference of Surfaces, 3D Curves, and Junctions from Sparse, Noisy 3D Data. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Volume 19, November 1997, Pages 1265-1277.

[HOP92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. Surface Reconstruction from Unorganized Points. *Computer Graphics (SIGRAPH '92 proceedings),* Volume 26, Pages 71-78, July 1992.

[MN99] A. Mangan and R. Whitaker. Partitioning 3D Surface Meshes using Watershed Segmentation. *IEEE Trans. on Visualization and Computer Graphics,* Volume 5, Pages 308-321, October 1999.

[MD00] G. Medioni, C. K. Tang and M. S. Lee3. *A Computational Framework for Segmentation and Grouping.* Elsevier Science B.V., 2000.

[MIL77] R. Millman and G. Parker. *Elements of Differential Geometry.* Prentice Hall, 1977

[ON83] B. O'Neill. *Semi-Riemannian Geometry with Applications to Relativity.* Academic Press, 1983.

[RS93] N.S. Raja and A.K. Jain. Obtaining Generic Parts from Range Images Using a Multi-view Representation. *CVGIP: Image Understanding.* Vol 60, Pages 44-64, July 1994.

[SC99] M. Soucy and D. Larendean. A General Surface Approach of a Set of Range Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Volume 17, Pages 344-358.

[ST86] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, 1986.

[TG96] C. K. Tang and G. Medioni. Inference of Integrated Surface Curve, and Junction Descriptions from Sparse 3D Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Volume 20, Pages 1206-1222, November 1996.

[TG99] C. K. Tang and G. Medioni. Robust Estimation of Curvature Information from Noisy 3D Data for Shape Description. *Proceedings ICCV 1999,* Greece, September 1999.

[TB95] G. Taubin. Estimating the Tensor of Curvature of a Surface from Polyhedral Approximation. *Proceedings of ICCV,* Pages 902-907, 1995.

[Wu97] K. Wu and M. Levine. 3D Part Segmentation Using Simulated Electrical Charge Distribution. *Transactions On Pattern Analysis and Machine Intelligence*, Volume 19, Pages 1223-1235, November 1997.

VITA

Born April 12, 1968, Tamara S. Bouma grew up in Firth, Nebraska with her two sisters and brother. She graduated from Norris High School in 1986, after which, she worked several odd jobs until joining the Air Force in 1990. In 1992 she left the Air Force to attend college in California. Attending American River Community College in Sacramento, CA from 1993 until 1995, she graduated with highest honors with an associates degree in Mathematics. She received a Regents scholarship to attend the University of California at Davis (UCD) where she graduated with a Bachelor's in Science Degree in Mathematics. While at UCD, she became a member of Golden Key National Honor Society, Pi Mu Epsilon, which is a mathematics honor society. She also received a Departmental Citation, and graduated with honors in 1997.

After graduating from UCD, she began working as a Software Engineer for IBM in Tucson, Arizona. She programmed a mainframe data availability program called Hierarchical Storage Manager until 1999. She moved to Seymour, Tennessee to pursue a Master's Degree in Mathematics at the University of Tennessee in Knoxville. Her primary interest in mathematics is geometry and its relationship to technology. This thesis is a reflection of this interest.

In 1993 she married Stephen Lyons. After finishing her master's degree in mathematics, she plans to move closer to her family in Springfield, MO with her husband.