6-1-2009

Surface Reconstruction through Time

LeeAnn T. Brash
Dartmouth College

Dartmouth Computer Science Technical Report TR2009-648

Surface Reconstruction through Time

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Master of Science

in

Computer Science

by

LeeAnn Tzeng Brash

DARTMOUTH COLLEGE

Hanover, New Hampshire

June 2009

Examining Committee:

(chair) Robert L. Drysdale III

Devin J. Balkcom

Daniel N. Rockmore

Brian W. Pogue, Ph.D.
Dean of Graduate Studies

Abstract

Surface reconstruction is an area of computational geometry that has been progressing rapidly over the last decade. Current algorithms and their implementations can reconstruct surfaces from a variety of input and the accuracy and precision improve with each new development. These all make use of various heuristics to achieve a reconstruction. Much of this work consists of reconstructing a still object from point samples taken from the object's surface.

We examine reconstructing an n-dimensional object and its motion by treating time as an $n+1$st axis. Our input consists of $n-1$-dimensional scans taken over time and at different positions on the original object. This input is mapped into $n+1$ dimensions where the $n+1$st dimension is a scaled time axis and then fed into an existing surface reconstruction algorithm. A cross section of the reconstructed surface perpendicular to the time axis yields an approximation to the shape of the n-dimensional surface at the corresponding point in time.

The intended application for this work is the reconstruction of medical images from scanning technology such as MRI or CT into moving $3d$ surfaces. We investigate reconstructing $2d$ moving surfaces through time as a preliminary step towards the moving $3d$ problem.

We spend most of our efforts in this thesis on the problem of computing a scaling factor for mapping time into the $n+1$st axis to minimize the number of scans needed to meet the sampling requirements for an existing surface reconstruction algorithm. We give three bounds, based on features of the $2d$ moving object, that are necessary to accomplish this.

Acknowledgments

Journeys often take us where we don't expect to go. I did not expect to finish this particular journey at this place but here I am. An education is never a solo journey and the following people are my boons.

I thank my adviser, Scot Drysdale, who has seen me through the best and the worst of this phase of my life and education. He has extended to me his caring and his support despite all the setbacks and he has celebrated with me the incremental triumphs that make this difficult road worth taking. He is a wonderful examplar of the spirit of Friendship brought into the academic world and I am blessed that he has shared that with me over these years.

My husband Sandy, my mother-in-law Ann, and my sister-in-law Sally picked me up when I came home discouraged, and they cheered for me when I made each breakthrough. They have sustained me over the last few years and opened their hearts and their home to me.

Last and most importantly, I thank my parents, Shih-Ying and Ju-Yu Tzeng. They came here to further their educations and they created a life here, and they are everything that I have behind me in every step I take.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Surface reconstruction from a cloud of points is a well-known area of computational geometry. A good deal of recent and current work focuses on extending two-dimensional methods to three dimensions, with varying degrees of success using various techniques [4, 5, 12]. As with many geometric problems, the two-dimensional case is far more straightforward than the three-dimensional case and the variety of heuristic approaches to the three-dimensional case reflects this. These results are useful in cases of reconstructing a static object.

Many applications of surface reconstruction require highly accurate reconstructions. For instance, a number of projects use surface reconstruction techniques and technology to record items of cultural importance both to preserve the items in digital form and also to further analyze them. [8, 18, 20]

A key area of current interest is reconstruction of medical images, primarily in three dimensions. [15, 21, 32, 36] Two dimensional images are already available to a medical specialist. The scans produced by MRI, CT or PET technology yield two-

dimensional slices of the object (usually an organ or region of a patient's body), and the specialist can simply look at the series of two-dimensional images without need of any reconstruction.

When wanting to view something in three dimensions, however, the scans are still two-dimensional images and collecting a series of them to capture a three-dimensional picture takes time to produce. Further, when scanning a human body, there is movement between each consecutive scan. Many current three-dimensional reconstruction techniques stack a set of scan slices together to reconstruct the object but do not take into account the movement between scans. A lot of other research focuses on aligning (or registering) different slices to get closer to an accurate reconstruction. The result is a rough approximation of the object's true shape. For some purposes, this is sufficient. For some other needs, though, the accuracy and precision of the reconstruction can be critical. For example, building a model of a generic human body does not require a particular kind of accuracy because the model does not represent an actual person. If the model needs to represent an actual patient, however, perhaps for diagnostic or surgical purposes, then it is easy to see why greater accuracy and precision in the reconstruction become very important.

1.1 Existing medical engineering approaches

Much of the work in medical engineering focuses on modeling and simulation using discrete meshes. Meshing is itself an area of extensive research. The goal of this work is primarily to better understand the biomechanical and biophysical behavior of the organs being modeled or to try to predict longer-term consequences such as tumor

growth. [30, 31, 32, 36]

Work has also been done on creating digital human models, both in the form of a generic "ideal" human model and in the form of creating smaller, localized digital models from real patient data. This includes work on generating the discrete meshes to represent parts of the human body. These models are either static or do not need to account for real-time motion because their goal is to see how things move in relation to each other.

These all approach the problem of modeling a human body and its organs from a perspective that assumes knowledge of the shapes in advance. However, none of this work addresses the question of whether it is even geometrically possible to build an accurate (moving) model that incorporates motion over time using only still scan data.

1.2 Background / Surface reconstruction

In computational geometry, PowerCrust, Cocone, and Eigencrust are just a few examples of the well-known implemented algorithms for surface reconstruction from point clouds. For our investigations, we use PowerCrust because it is simple and has a clear set of conditions under which it is guaranteed to return a geometrically and topologically correct approximation to the original surface [1]. In particular, PowerCrust simply requires a sufficiently dense sampling of the surface.

To specify "sufficiently dense," we first need to define the medial axis. The medial

[1]PowerCrust's theoretical arguments use a closed, bounded, smooth 2d surface, but the authors report "good empirical results on inputs including models with sharp corners, sparse and unevenly distributed point samples, holes, and noise, both natural and synthetic." [4, 5]

axis is the closure of the set of points that have two or more closest points on the original closed surface. The medial axis can contain points both inside and outside the closed surface, depending on the shape of the surface.

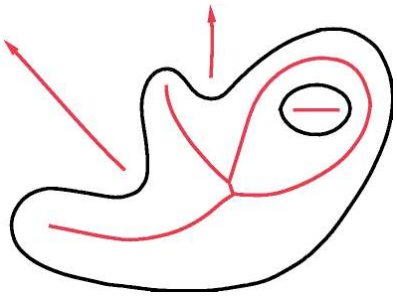

Figure 1.1: Two closed curves and their combined medial axis. Image from Amenta, Bern and Kamvysselis [3].

The distance from a point on the surface to the medial axis is known as the local feature size, or LFS. Intuitively, the LFS is a measure of the size of a detail, or feature, on the surface. A part of the surface that has smaller features needs more sample points to capture the details, whereas a part of the surface that has only a large feature (i.e., no small features and is thus relatively smooth) requires fewer sample points to capture its contours.

A sample of points is called an r-sample when any given point of the surface has a sample point no farther away in distance than r times its local feature size. For small enough r, the sample is then considered to be sufficiently dense.

Given a sample of points that meets this condition, PowerCrust is guaranteed to return a good reconstruction of the original surface.

1.3 Need and motivation

With the application area we have in mind, namely medical image reconstruction, we are no longer dealing with still shapes and we are definitely dealing with three dimensional objects. Let's look at scanning a heart as an example.

A heart beats at least once a second, and is generally in constant periodic motion. With current and foreseeable technology, we cannot scan the whole heart at once. We can only take a picture of one $2d$ slice at a time and by the time we take the second slice, even if it is very close physically, the heart has moved.

Charting this over time, we have a series of slices from different points in the heart's motion.

What we know is that the heart's motion is periodic. What we also know is that between scan slices, the motion is continuous in the analytical sense (more or less). Ideally, the distance between the slices is both small in time and small in space. In other words, we would like to have enough scan pictures to cover as many slices of the heart as possible, and we would like to have these pictures taken frequently enough to capture the motion of the heart.

Even when scanning something that isn't as active as the heart, it is practically impossible to align two consecutive scan slices simply because the body is always moving. As an example, when scanning the lungs, one frequently-used technique is to ask the patient to hold in a deep breath. When it is time to take another scan, the patient is asked to hold in a deep breath again. While this gets the two lung positions very, very close, it is still not exact, and at the resolution of the scanning technology, these differences can be significant.

It is therefore important from a surface reconstruction perspective not only to capture the shape of the surface in three-dimensional space but also to capture the shape of the three-dimensional object as it changes over time.

Clearly, there is a need for accurate surface reconstruction in $3d$ plus its motion in time, t. We call this $3d + t$. At present, we know that it is possible to take still $2d$ slices and reconstruct them into a still $3d$ object. What we don't yet know is whether we can take a series of still $2d$ slices and accurately reconstruct the $3d$ object and its motion through time.

Is it possible to reconstruct the shape of a surface over the course of some motion by filling in missing data by looking at where the nearby/neighboring points go at the next step in time? Further, is it possible to consider this problem as a four-dimensional surface reconstruction, where time acts as the fourth dimension?

To test these ideas, we need a four-dimensional surface reconstruction algorithm. While this is known to be theoretically possible [13], there has been limited motivation to implement such an algorithm in dimensions higher than three. For our investigations, we first need to know if treating time as an additional dimension is even possible. In this thesis, we will explore the $2d+t$ problem so that we can evaluate whether the $3d + t$ problem even makes sense to pursue.

Chapter 2

Our goals

The primary problem at hand is how to reconstruct the motion of a $2d$ object through time, given data from scans taken as the object moves.

Because the scans in three dimensions are planar slices, the analogous scans in two dimensions must be linear slices. Scans are taken using a sweepline technique. In the case of a $2d$ object, this means a line moving across a plane. Where the line intersects the boundary of the object is recorded as a data point and the time of the scan is also recorded. The resulting collection of data points is a series of intersections on the plane stacked in order of the time taken.

We want to see if $2d + t$ can also be viewed as a three-dimensional reconstruction by mapping t into the z-axis. In particular, we want to cast the $2d + t$ problem into a $3d$ problem and then use the existing $3d$ reconstruction techniques to achieve our desired results. The obvious first question is how to map t into the z-axis.

As long as the t-axis is handled in such a way that the conditions for the existing $3d$ surface reconstruction methods are satisfied, this ought to be possible. Further,

taking cross-sections along the t-axis of the resulting reconstruction ought to yield the shape of the object at the corresponding point in time.

More precisely, we want to somehow map the time-axis into the z-axis and feed the resulting set of points into PowerCrust. The collected points take the form $f(t) = (x(t), y(t), t)$, where t is time and $(x(t), y(t))$ is the location of a point from the object in its native plane at time t. In other words, $f(t)$ is a subset of the locus of the object as it changes over t.

As mentioned before, the obvious first hurdle is to determine how to map t into the z-axis. Specifically, we need a *scale* by which to map t into the z-axis. We need to find a constant c such that we can plot $g(t) = (x(t), y(t), ct) = (f_x, f_y, cf_z)$.

For the purposes of our work, we have only the detected points of intersection between the object and the scanline. We want to find c such that PowerCrust returns a recognizable reconstruction from the scaled collection of points.

It is important to note that once a c has been determined, the scanning scheme must also be determined to achieve the required r-sample for PowerCrust or whichever reconstruction algorithm one wishes to use. While we could theoretically increase the sampling density to an extreme for a very small r, in reality this is both time-consuming and less practical.

Toward these ends, we look to PowerCrust's sampling requirement, which is based on local feature size. The $2d$ object already has a sampling density requirement based on local feature size. If the scaling factor for our time axis creates a smaller local feature size in $3d$ than the smallest local feature size in $2d$, then we need a denser sample than in $2d$. In the scaling of our time axis, we want to avoid requiring a

greater sampling density for our $2d + t$ (i.e. $3d$) reconstruction.

Thus, we seek a scaling factor, c, for the time axis such that the reconstructed $3d$ minimum local feature size preserves the $2d$ minimum local feature size as closely as possible.

Chapter 3

Experiments

3.1 Setup

The purpose of our experiments was to see how the scaling factor affected the final reconstruction in $2d + t$. To this end, we ran seven sets of experiments, each with a different shape or a different type of motion. We placed all of the experiments within a bounding box and ran a sweepline-type scan, which we will call the *scanline*.

The scanline moved back and forth across the bounding box starting at time t_0. For each t_i, we recorded the points of intersection (x, y) between the scanline and the surface in the form $(x, y,$ scaled-$t_i)$. For each set of experiments, we had three main parameters: duration of the period of the scanline, number of scans per period of the scanline, and t-scale, or c. We call the period of the scanline the *scanperiod*. For each pairing of scanperiod and scans per scanperiod, we ran a wide range of t-scales since that was our first point of interest. The scanperiod was first defined in terms of periodic motion; specifically, we used a parameter called *scanratio* that was the ratio

between the duration of the object's period of motion and the scanperiod.

To get preliminary numbers, we used a scanratio of 0.05 and a scanrate of 100. One full period of motion, where periodic motion was used, took place in 2000 scans. We treated each scan as taking one time unit.

For each experiment, except where noted, we collected 6000 scans, i.e., up to t_{6000} or three periods of motion. The point cloud formed by these points was then fed into PowerCrust ([4]). We visually inspected the resulting reconstructions and also checked the cross-sections parallel to the xy-plane.

We begin by explaining each experiment. The results and analysis follow.

3.1.1 Basic ellipse

To get an idea of how this worked in practice, we started with a basic smooth shape and a simple periodic motion. We used an ellipse so that the shape had some identifiable feature and rotation as the motion. For this experiment, the ellipse had a major axis of 6 and a minor axis of 4. The ellipse and its rotation were both centered at the origin. To make sure that the scanline covered the whole shape, no matter its current rotation, we used a bounding box of 6.4, also centered at the origin. In this case, each scan took place at some time t_i and produced at most two points of intersection on the xy-plane.

3.1.2 Bean in rotation

Because an ellipse is a very simple shape with only very basic features, we next tried the classic bean (Figure 3.1) from the original Crust paper [2]. It is nearly as

simple as the ellipse but it has one interesting feature, namely the concave divot. This feature has both a smaller minimum local feature size and a concave curvature, both of which can be missed in a poor reconstruction.

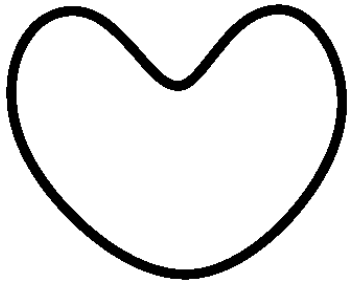

Figure 3.1: The classic "bean" shape.

For our purposes, we modeled the bean as a union of a two half-circles and two half-ellipses. We centered the rotation at the center of the two concentric half-circles. The larger circle had a radius of 3, the smaller circle had a radius of 0.5, and the two ellipses had major axes of 2.5 and minor axes of 1. Again, we used a bounding box of 6.4 to be certain of scanning the whole shape in any position of rotation.

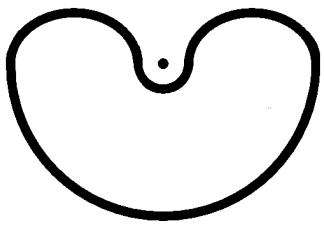

Figure 3.2: The bean modeled as two concentric half-circles and two half-ellipses. The center of the two circles is the origin.

3.1.3 Shrinking and expanding ellipse

Both of the previous two experiments used a still shape in motion. To see what happened in a changing shape, we started again with an ellipse and this time made it shrink and expand. We used a sine function to control the growth of the motion. The center of scaling was kept at the origin. The largest ellipse had a major axis of 3 and a minor axis of 2. The smallest ellipse had a major axis of $\frac{3}{4}$ and a minor axis of $\frac{1}{2}$. The bounding box was again 6.4.

3.1.4 Shrinking and expanding bean

Again, we wanted to see if this would still work with a slightly more complicated shape. In this case, we used the classic bean and kept the union of the four pieces we used before to represent the bean. As with the ellipse, we used the origin as the center of scaling.

3.1.5 Bean with sliding divot feature

To look at what happened with a more interesting type of motion, we took the classic bean and moved the feature back and forth. This made the shape itself change features and feature locations, and it allowed us to look at a change in surface normals to see how that might affect the reconstruction efforts.

We used the same model of the bean as in our previous experiments but slid the inner half-circle back and forth, again using a sine function to control the motion of the divot. The extent of the motion was bounded by keeping the minimum local feature size no smaller than that of the divot itself.

3.1.6 Translating circle

To look at a much simpler example, we translated a circle along the y-axis. For this experiment, we used a scanrate of 200. The circle moved 20 units along the y-axis in 12000 scans (i.e. 12000 time units).

3.1.7 Circle rotating about a tangent point

For this experiment, we placed a circle of radius 1 tangent to the origin and revolved it around the origin.

3.2 Experimental Results

In each set of experiments, PowerCrust produced both the correct shape and the correct motion for a range of scaling factors for the t-axis. The cross-sections showed that the $2d$ shapes at each t_i were also correct. As expected, given a scanning scheme, the range of scaling factors that yielded good reconstructions varied depending on both the shape and the motion. Table 3.1 lists the results.

Table 3.1: Experimental Data

Shape	**Motion**	**min $2d$ LFS**	**c range**
Ellipse	Rotation	$\frac{4}{3}$	0.002 - 0.09
Bean	Rotation	0.5	0.004 - 0.01
Ellipse	Shrinking/Expanding	$\frac{1}{3}$	0.002 - 0.02
Bean	Shrinking/Expanding	0.5	0.002 - 0.01
Bean	Sliding divot	0.5	0.002 - 0.008
Circle	Sliding	1.0	less than 0.0001 - 0.04
Circle	Revolving	1.0	0.002 - 0.03

What we observe is that the minimum local feature size seems to figure into the scaling factor. Between the rotating ellipse and the rotating bean, the only real difference is the minimum local feature size. We see a difference in the range of good scaling factors, which indicates that something other than the motion contributes to calculating a good scaling factor.

Also, looking at the ellipse in two different kinds of motion, we see that the good scaling range is different. The ranges overlap, though for the shrinking and expanding motion, the range spans a smaller interval. We see a similar situation for the rotating bean and the shrinking and expanding bean. This warrants looking at more experiments, as we have done.

The translating circle is discussed more thoroughly in the analysis below.

In varying the scanperiod and the scanrate, we also see that some reconstructions are better and some are worse. This leads to some intuition about how we might optimize scanning. Once we have a good scaling factor, we can potentially calculate a good scanperiod and scanrate to meet PowerCrust's sampling requirements. We discuss this further in section 3.3.

From the experiments, we see that, given a scanning scheme, the factor by which the time axis is scaled is strongly tied to the recognizability of the overall reconstruction. If the time axis is too wide, then we do not have sufficient sampling to identify the shape vertically to show the whole $2d$ object (Figure 3.8). If the time axis is too tight, then we do not have enough sample points to distinguish between the correct surfaces and subsequent phases of motion, e.g., during successive rotations. This creates "false" surfaces between farther points in time (Figure 3.9).

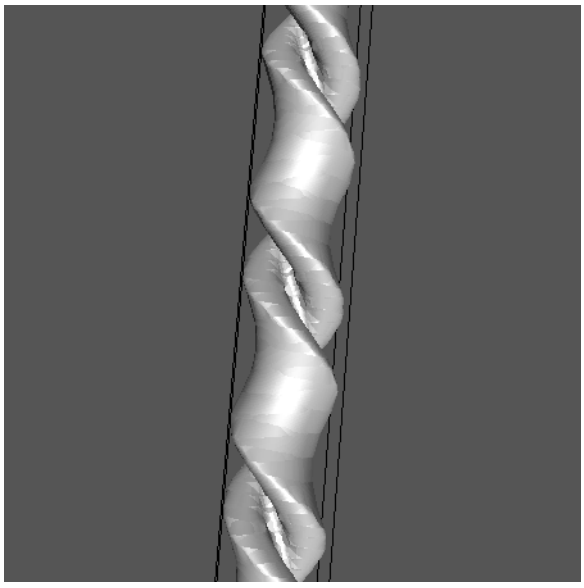

Figure 3.3: Rotating bean.

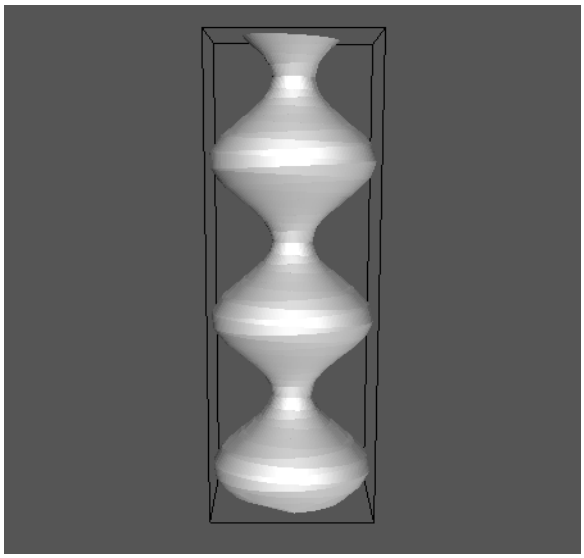

Figure 3.4: Shrinking and expanding ellipse.

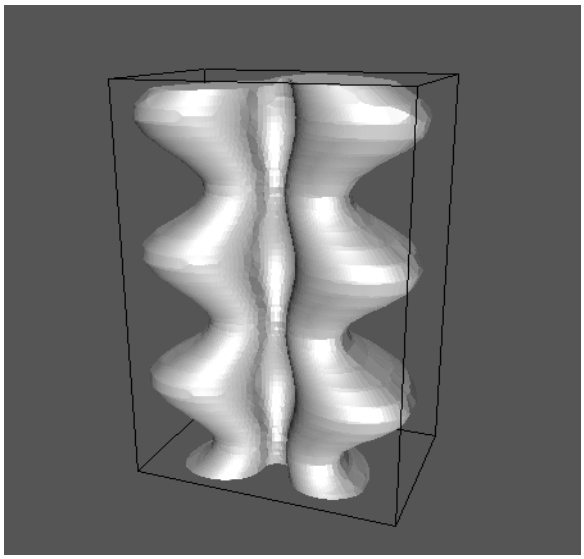

Figure 3.5: Shrinking and expanding bean.

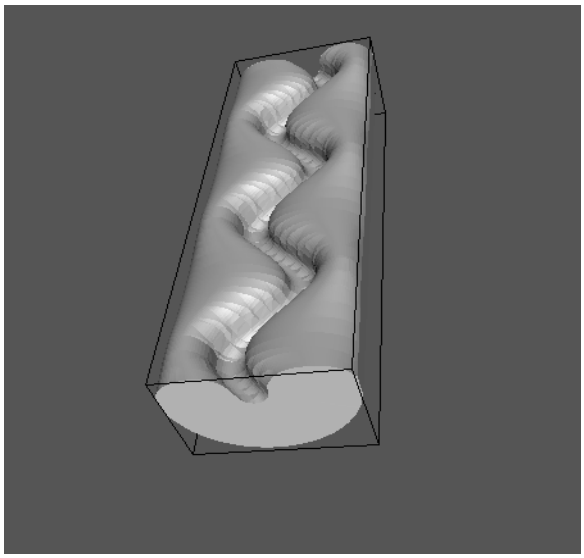

Figure 3.6: Sliding divot on the bean.

We have seen that, given a particular scanning scheme, we can find a range of scaling factors that yields a recognizable reconstruction for each experiment (Figures

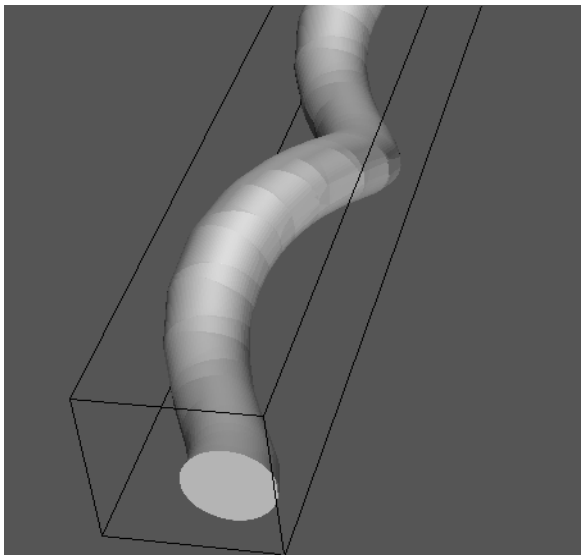

Figure 3.7: Circle rotating about a tangent point.

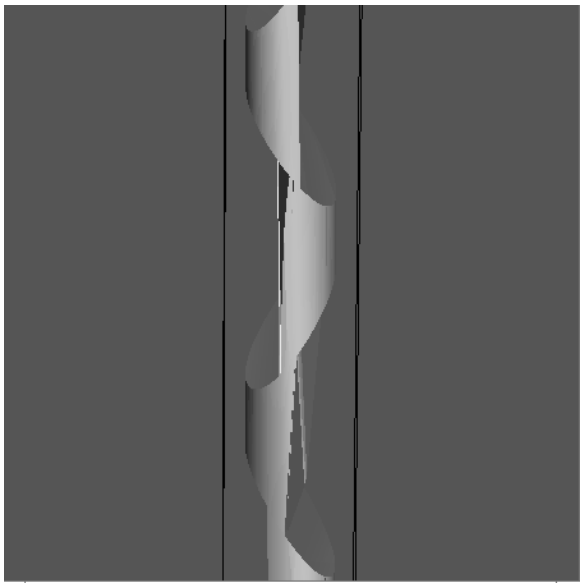

Figure 3.8: Rotating ellipse: When the scaling factor is too large, we lose surfaces in the reconstruction.

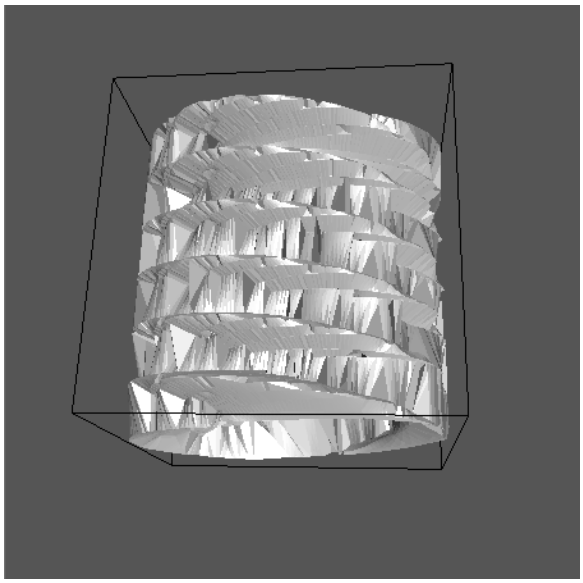

Figure 3.9: Rotating ellipse: False surfaces appear when the time axis is scaled too tightly.

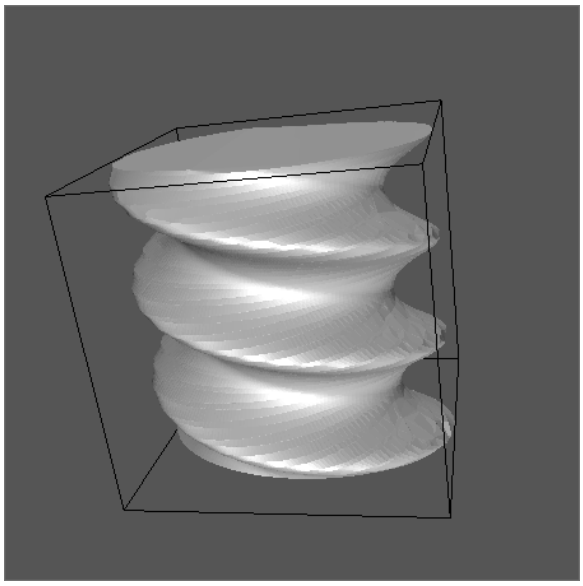

Figure 3.10: Rotating ellipse: The reconstruction works when the scaling factor is within the "just right" range. This is a smaller scaling factor that works for the rotating ellipse.

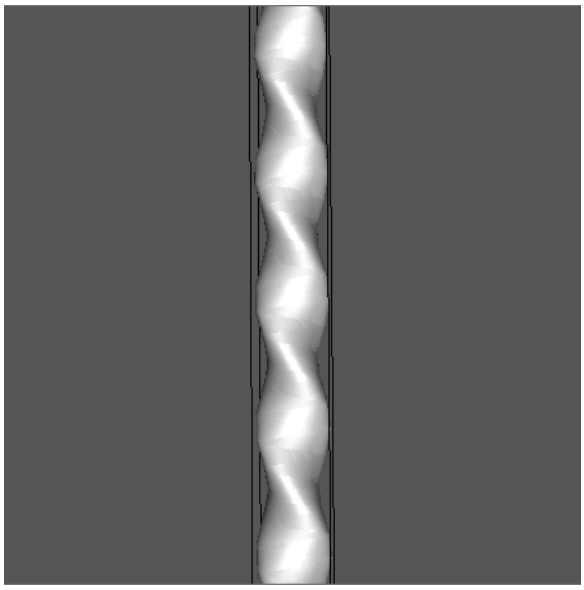

Figure 3.11: Rotating ellipse: The reconstruction also works when the scaling factor is on the larger end of the "just right" range.

3.10, 3.11). This range varies by shape and by motion. As suspected, the size of local features of the $2d$ object seems to affect the range of usable scaling factors. Also as suspected, the motion of the object affects the range of scaling factors. Similar to the size of features of the object, the size of the "features" of the motion appears to constrain the scaling factors.

While feature size of a surface has been well-defined for these purposes by Amenta, Bern and Eppstein [2], feature size of motion is a new notion in surface reconstruction. So far, it appears that acceleration and velocity are two primary elements in determining a useful scaling factor.

We thus explore the relationship between a good scaling factor, the minimum $2d$ local feature size and the motion. We can break down the motion of the object into its velocities and accelerations. This is explained further as we discuss how to calculate

a good scaling factor in chapter 4.

3.3 Sampling rate

Before we discuss the scaling factor in more detail, it is important to note that both the scanperiod and the scanrate affect how well our moving shapes are reconstructed. We know that PowerCrust has a minimum sampling requirement, but because our $3d$ surface is not known in advance and thus we do not have the $3d$ local feature sizes, we have to determine our sampling using a good guess about the local feature sizes. This is where the t-axis scaling factor can help us.

From the experimental results, we know that a very large scaling factor requires more scanperiods to yield a smooth reconstruction. A very small scaling factor requires more scans per scanperiod, or a higher scanrate. Finding a scaling factor that balances the required scanning densities of both the scanperiods and the scanrates allows us to maximize the efficiency of the scanning.

It is worth noting that our experiments consist of relatively "round" shapes within a bounding boxes that are relatively "square." None of the shapes are extremely long and skinny, and we do not have any part of a shape's boundary that approximates a primary axis. This has allowed us to avoid a particular sampling difficulty that arises when a long, skinny shape (or a significant part of a shape's boundary) lies parallel to the scanline (Figure 3.12). Fortunately, an easy solution to these situations is to rotate the primary axes a little bit (e.g., using perturbation, Figure 3.13). This moves the boundaries away from being parallel to the scanline and reduces the opportunities for missing the boundaries during scanning.

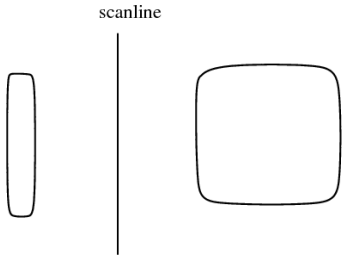

Figure 3.12: The scanline can easily miss the parts of the boundaries that are parallel to it, both for a long, skinny shape and for boundaries of non-skinny shapes that run parallel to the scanline.

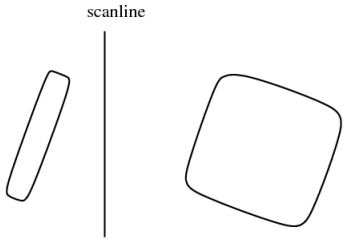

Figure 3.13: We can perturb the scanning area relative to the scanline to minimize problems from boundaries being parallel to the scanline.

We already have a sampling density requirement based on the original $2d$ shape. Without knowing the local feature size at specific points on the shape, we can still use the minimum local feature size to choose the r for guaranteeing an r-sample for PowerCrust. Ideally, our scanning scheme will not require a significantly denser sample to reconstruct the shape and its motion. Thus, we use the $2d$ minimum local feature size as a starting point for a proposed lower bound on the $3d$ minimum local feature size and thus also on the scaling factor.

Chapter 4

Analysis

We need to choose a scaling factor for the time axis before we can input our experimental data into PowerCrust. We already know that the PowerCrust-like algorithms require a minimum sampling density based on the local feature size of the area being sampled, or in other words that every point on the surface of the object must be "close enough" to a sample point [5].

An immediate observation is that the minimum local feature size of the $3d$ reconstruction can be smaller than the local feature size of the original $2d$ object but cannot really be bigger. If the minimum local feature size of the $3d$ reconstruction is bigger than the $2d$ minimum local feature size, then the overall minimum local feature size is defined by the minimum local feature size of the original $2d$ object. Only if the $3d$ minimum local feature size is smaller than the $2d$ minimum local feature size does the $3d$ minimum local feature size overtake the $2d$ minimum local feature size as a component in determining the scaling factor.

Our original idea had been to limit the $3d$ minimum local feature size to be at least

as big as the $2d$ minimum local feature size. This is actually impossible in the case of non-accelerating motion with simple velocity, as in the case of our translating circle experiment. Consider a circle that is moving (in translation) at constant velocity. (Figure 4.1) The space defined by the circle as it moves is an elliptical tube. (Figure 4.2) The only way to get the tube to preserve the minimum local feature size of the circle is to map t to z using a scale of $c = \infty$, which would give us a circular tube. Otherwise, the tube narrows if we use any $0 < c < \infty$ to map t to z.

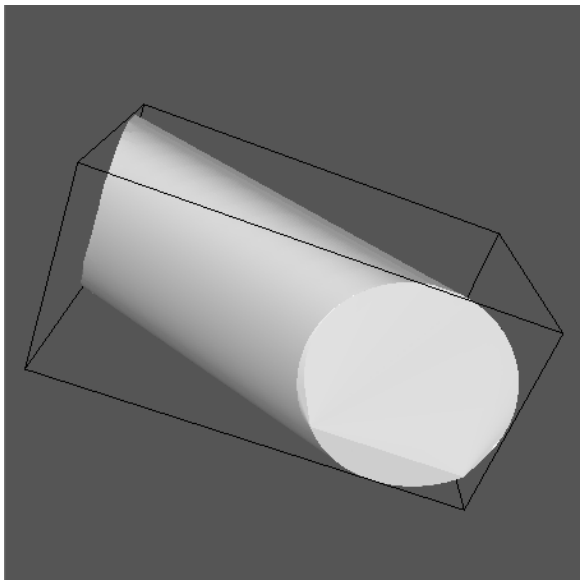

Figure 4.1: A translating circle viewed from the bottom of the xy-plane.

This means that we cannot preserve the $2d$ minimum local feature size in the $3d$ reconstruction. Instead, we try to keep the $3d$ local feature size no smaller than a constant fraction of the $2d$ local feature size. By selecting a factor, κ such that $0 < \kappa < 1$, we can continue calculating c based on κ times the $2d$ minimum local feature size. We can easily see that the higher the velocity the more eccentric the

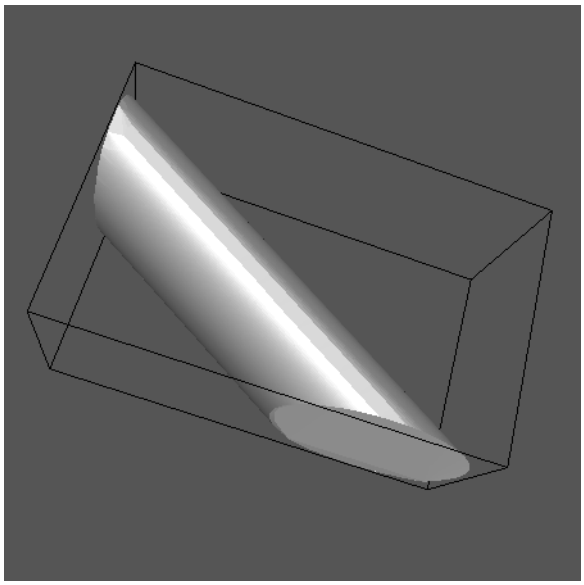

Figure 4.2: The same translating circle viewed from the side (yz-plane).

elliptical cross-section of the tube and therefore the smaller the $3d$ minimum local feature size.

Thus we see that the maximum velocity of any part of the surface is a necessary component in calculating a bound on our scaling factor c.

4.1 Velocity

We now look at the circle example more closely. The translating circle forms an elliptical tube when mapped through time into the z-axis. Note that, given a semi-major axis of a and a semi-minor axis of b, the minimum local feature size in an ellipse (assuming $a \geq b$) is $\frac{b^2}{a}$. In this situation, a is the radius of the original circle and b is half the width of the elliptical tube. Thus, our $3d$ minimum local feature size is $\frac{b^2}{a}$. If

we want the $3d$ minimum local feature size to be at least a factor κ $(0 < \kappa < 1)$ of the $2d$ minimum local feature size, which we will now call L, then we have the following:

$$\kappa L = \frac{b^2}{a},$$
$$\kappa L a = b^2,$$

and

$$\sqrt{\kappa L a} = b.$$

Further, we know that $a = L$ by definition, so that

$$\sqrt{\kappa L L} = b,$$
$$\sqrt{\kappa L^2} = b,$$

and

$$L\sqrt{\kappa} = b.$$

Now, let the velocity vector serve as our distance axis, which will be in the xy-plane. Then t will be mapped into the z-axis, with a scale c to be determined. The ratio between a and b can be placed into a trigonometric relationship as a sine. (Figure 4.3)

$$\sin\alpha = \frac{b}{a},$$
$$\sin\alpha = \frac{L\sqrt{\kappa}}{L},$$
$$\sin\alpha = \sqrt{\kappa},$$

or

$$\alpha = \arcsin \sqrt{\kappa}.$$

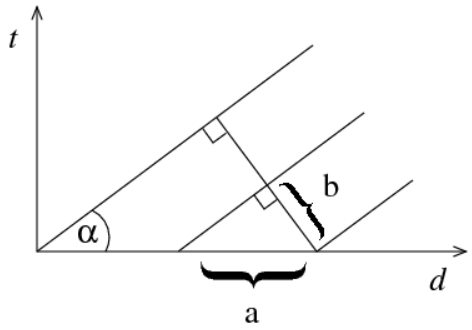

Figure 4.3: The relationship between the semi-major axis a and the semi-minor axis b of the elliptical tube is $\sin\alpha = \frac{b}{a}$. Distance, d, traveled in the y-direction takes time in the t-axis. To get the α we need as defined by the κ we choose, we scale t by some c. $\tan(\alpha) = ct/d = c/v$.

We already know that velocity $v = d/t$. So, to obtain the κ (and therefore α) that we want, we must scale t by some constant c. Taking another trigonometric relationship, we have that $\cot(\alpha) = d/ct$. Alternatively, we can take $\tan(\alpha) = ct/d = c/v$. So from κ, we get α, which gives us c and satisfies the following inequality.

$$\begin{aligned} c &\geq v \tan\alpha \\ &\geq v \tan \arcsin \sqrt{\kappa} \\ &\geq v \frac{\sqrt{\kappa}}{\sqrt{1 - \sqrt{\kappa}^2}} \\ &\geq v \sqrt{\frac{\kappa}{1-\kappa}} \end{aligned}$$

4.2 Acceleration

Similarly, when some part of the surface changes direction or speed, then the motion maps out as a curve in three dimensions. If the curve is tighter, then its local feature size is smaller. The relationship between the acceleration and the three-dimensional curvature indicates that we might be able to calculate a scaling factor such that the curvature does not give rise to a local feature size that is too small in three dimensions.

Our intuition is that acceleration does not face the same problem as the velocity tunnel. Specifically, while the velocity tunnel makes it impossible to preserve the $2d$ minimum local feature size as the overall $3d$ minimum local feature size, we have found no such direct counterexample for acceleration. Because acceleration always maps into a curve, we believe that this will very likely allow us to preserve the $2d$ minimum local feature size when acceleration is the constraining motion.

To consider this more thoroughly, we analyze the curvature of our mappings.

In differential geometry, the radius of curvature at a point is the radius of the osculating circle at that point. In computational geometry, this is equivalent to a maximal ball centered at an endpoint of the medial axis. Instead of the maximal ball that touches the shape at opposing boundaries, it is the maximal ball that sits tightly inside a curve in the boundary of the shape and defines the LFS of a point in that tight curve. (Figure 4.4)

To try to find a bound for our scaling factor using acceleration , we look at the relationship between the radius of curvature and acceleration. Greater acceleration means a faster change in velocity which in turn means a tighter curve. A tighter curve

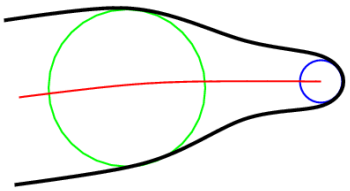

Figure 4.4: Two types of maximal balls. One is centered at an endpoint of the medial axis and nests into a tight curve of the surface. The other is centered at a non-endpoint of the medial axis and is tangent to (at least) two points on the surface.

has a smaller radius of curvature and thus a smaller LFS, so we focus on maximum acceleration.

In our experiments, we know the maximum acceleration of a point in our shapes because we selected the motions. By definition, the vector acceleration of a curve is the sum of the tangential acceleration and the normal acceleration. Tangential acceleration is the same as scalar acceleration, and normal acceleration is equivalent to centripetal acceleration. We can compute these components for our experimental shapes and motions.

$$\vec{a} = a_{scalar}\vec{\mathbf{T}} + a_{normal}\vec{\mathbf{N}}$$

We can rewrite this in terms of a function $f(t)$ as follows.

$$\vec{a} = f''(t)\vec{\mathbf{T}} + \frac{f'^2(t)}{R}\vec{\mathbf{N}}$$

For our experiments, we have the maximum accelerations shown in Table 4.1.

If we look locally at the point of maximum acceleration, we can rotate the coordinates to make that point a local minimum for a $1d$ function (Figure 4.5). This is because curvature is invariant under a change of orthogonal axes, such as rotation or translation or both. We then have a curve that we can frame in terms of $d = f(t)$.

Table 4.1: Maximum accelerations for each experiment

Shape and motion	**Scalar acceleration**	**Normal acceleration**
Rotating ellipse	0	$\frac{3\pi^2}{1000^2}$
Rotating bean	0	$\frac{3\pi^2}{1000^2}$
Shrinking ellipse	$\frac{1.125}{2000^2}$	0
Shrinking bean	$\frac{1.5}{2000^2}$	0
Sliding divot bean	$\frac{1}{2000^2}$	0
Revolving circle	0	$\frac{2\pi^2}{1000^2}$
Sliding circle	0	0

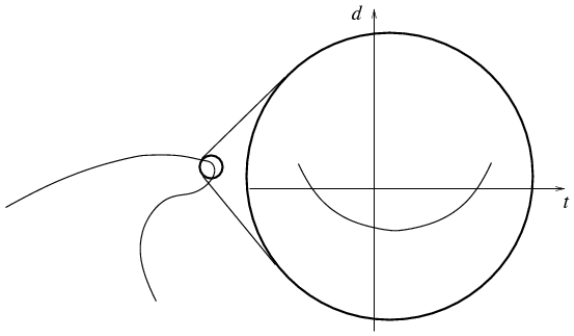

Figure 4.5: We can look at a point of maximum acceleration as a local minimum in local coordinates.

We can use the equation for the curvature of such a function to compute a scale for our t-axis. The radius of curvature for this function is defined as

$$\mathbf{R} = \frac{(1 + f'^2(t))^{\frac{3}{2}}}{f''(t)}.$$

We can control the tightness of this curve by scaling t. Each point is $(t, f(t))$, and scaling t by constant c, each point becomes $(ct, f(t))$. The new speed is thus

$$\frac{f(t)}{ct} = \frac{\text{original speed}}{c}$$

and acceleration is

$$\frac{\text{new speed}}{ct} = \frac{\text{original acceleration}}{c^2}.$$

Putting this into the equation for radius of curvature, we have

$$\mathbf{R} = \frac{(1 + (\frac{f'(t)}{c})^2)^{\frac{3}{2}}}{\frac{f''(t)}{c^2}}.$$

We want $\mathbf{R} \geq \kappa L$, where L is the minimum $2d$ local feature size.

$$\begin{aligned}
\mathbf{R} = \frac{(1 + (\frac{f'(t)}{c})^2)^{\frac{3}{2}}}{\frac{f''(t)}{c^2}} &\geq \kappa L \\
\frac{(1 + (\frac{f'(t)}{c})^2)^{\frac{3}{2}}}{\frac{f''(t)}{c^2}} \geq \frac{1^{\frac{3}{2}}}{\frac{f''(t)}{c^2}} &\geq \kappa L \\
\frac{1}{\frac{f''(t)}{c^2}} &\geq \kappa L \\
\frac{c^2}{f''(t)} &\geq \kappa L \\
c^2 &\geq \kappa L f''(t) \\
c &\geq \sqrt{\kappa L f''(t)}
\end{aligned}$$

Let us use the maximum value of $f''(t)$. We now have a way to find a c given κ, L and maximum acceleration.

4.3 Change in surface normal

Velocity and acceleration clearly govern the motion of the surface, especially when the size of the motion is larger. When looking closely at the surface, however, these

maxima may not be reached due to the small, local distances involved. To consider this type of situation, we examine local motion.

We take a point p that lies on the $2d$ surface. p has a maximal ball tangent to it. p also has a κL-ball tangent to it and contained in its maximal ball. This κL-ball can only touch the surface at p because it is smaller than the smallest maximal ball of any point on the surface, by definition. We look only at the points on the surface that are a distance of $2\kappa L$ or less from p, and we consider all motion relative to p. In other words, we fix p as our point of reference and we can then treat all of the local motion as rotations about p. We want to see what happens as the surface rotates about p.

The rate of rotation, or angular velocity ω, is the rate of change in the direction of the unit normal vector at p. The fastest rotation correlates to the fastest change in the direction of the normal vector at p. We will use this to find a bound for c. To do this, we look at how close points can be around p and therefore how small an angle of rotation can bring these points close to each other as mapped into time.

Let us look at a point p whose closest point on the $2d$ medial axis is an endpoint of the medial axis. The tightest positioning of the points adjacent to p is wrapping around the L-ball tangent to p and on the same half-circle as p. The points cannot wrap onto the other half-circle because they would have to come closer than the $2d$ local feature size allows. Let the surface then follow the tangent of its contact with the L-ball. This is the tightest arrangement of points possible in $2d$ due to the definition of local feature size.

Let us also bring the $2\kappa L$-ball of p into the picture. The surface, as it extends

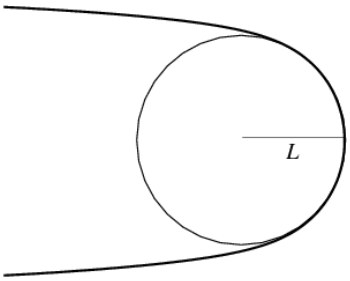

Figure 4.6: Close "wrapping" of points along an L-ball.

away from p, intersects the $2\kappa L$-ball at two points, x and y, that are a minimum distance of $2L$ from each other. If this part of the surface is rotating around p at an angular velocity of ω, then x might move into the κL-ball of y (or without loss of generality, vice versa) if ω is sufficiently large. Fortunately, this type of situation is part of the analysis for velocity, above.

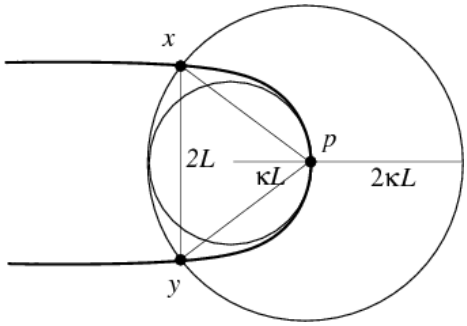

Figure 4.7: Distal points of local motion fall under velocity-based constraints.

What remains is to examine what happens at the center of rotation where the surface twists the most. For this, we look to the helicoid to model such behavior. A helicoid can be parametrized as follows. $x = u\cos\theta, y = u\sin\theta, z = \gamma\theta$. This works well to model a rotating section of the surface as it moves through time, as mapped

into the z-axis. γ already scales the z-axis of the helicoid, much as we map motions into the t-axis with scaling factor c. This is very useful right away.

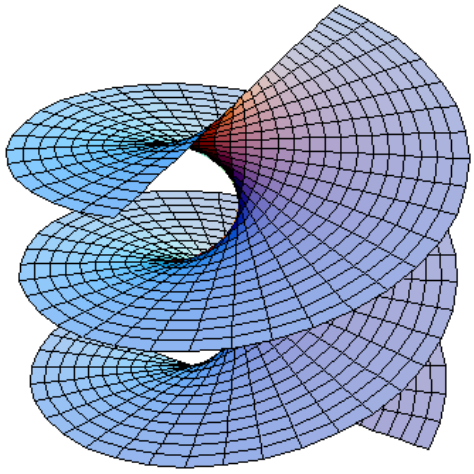

Figure 4.8: A helicoid. [35]

The angular velocity of the helicoid is $\omega = \dfrac{\theta}{\gamma\theta} = \dfrac{1}{\gamma}$. This is also the rate of change of the unit normal vector. By definition, the smallest local feature size on a helicoid is at the center of rotation, and it is defined as γ. Thus, if we want to keep the local feature size in $3d$ at least as large as κL, we choose a scaling factor c such that $c\gamma \geq \kappa L$, that is $c \geq \dfrac{\kappa L}{\gamma}$. In the case of the maximum rate of change in unit normal vector, that is the maximum angular velocity, we have $c \geq \kappa L \max \omega$.

Chapter 5

Proofs

We now show that our proposed bounds on c lead to feasible bounds on the $3d$ local feature size of the reconstructed surface through time. We are given $L = 2d$ minimum local feature size, maximum velocity, maximum acceleration and maximum rate of change in surface normal. We also choose κ such that $0 < \kappa < 1$. From this, we determine the constraints for c.

To show the validity of our proposed bounds on c, we observe that the local feature size of the surface in $2d$ and its motion are limited both by points that are nearby and by points that are farther away on the original $2d$ surface. We call these *adjacent* and *non-adjacent* points with respect to a point p on the surface. Adjacent points are within $2\kappa L$ of p. Non-adjacent points are all the points on the surface that are farther than $2\kappa L$ from p.

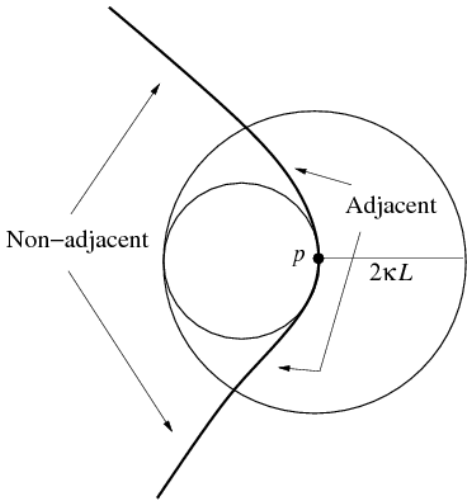

Figure 5.1: Adjacent vs. non-adjacent points on a surface.

5.1 Velocity

We first consider the motion of two non-adjacent points on the surface and the smallest possible distance between them. We will show that if a non-adjacent point enters the κL-ball of a point p_j, then its velocity must be greater than the maximum velocity. Thus the non-adjacent point can not enter the said κL-ball if we have selected a qualifying scaling factor c.

Suppose we have a point p_i at time t_i within a κL-ball that is tangent to the $3d$ surface at some point. p_i has a motion $p_i(t)$ that brings it to its location at time t_i. Without loss of generality, let us say that the κL-ball is tangent to the surface at a point p_j and that p_j is in the plane corresponding to time t_j.

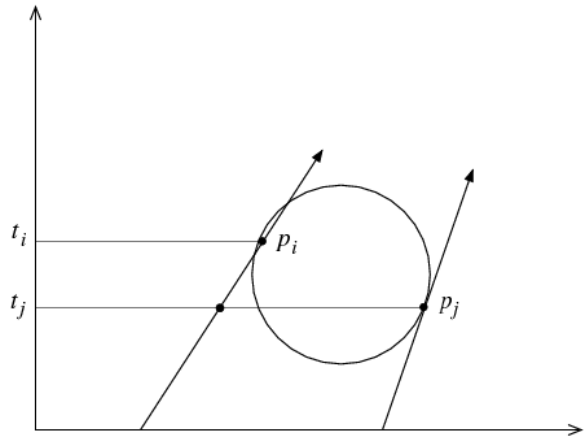

Figure 5.2: Points moving near a κL-ball.

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} < 2\kappa L$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 < 2^2\kappa^2 L^2$$

$(z_i - z_j)^2 > 0$ because otherwise, we would break L in the $2d$ plane at time $t_i = t_j$.

$$(x_i - x_j)^2 + (y_i - y_j)^2 < 2^2\kappa^2 L^2 - (z_i - z_j)^2$$

$|z_i - z_j|$ is the scaled difference in time. So $|z_i - z_j| = c\Delta t$. $c \geq v\sqrt{\frac{\kappa}{1-\kappa}}$ and $c \geq \sqrt{\kappa L a}$, where a is the maximum acceleration and v is the maximum velocity. Thus $c \geq v\sqrt{\frac{\kappa}{1-\kappa}}$, and so $|z_i - z_j| \geq v\sqrt{\frac{\kappa}{1-\kappa}}\Delta(t)$, or $\Delta(t) \leq \frac{|z_i - z_j|}{v\sqrt{\frac{\kappa}{1-\kappa}}}$.

The distance traveled by $p_i(t)$ in time $\Delta(t)$ is at least $2L - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. This is because at time t_j, $p_i(t)$ can be no closer than $2L$ to p_j. At time t_i, $p_i(t)$ has traveled to the position of p_i. So we have a minimum speed of

$$\frac{2L - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\Delta(t)}$$

.

$$\frac{2L-\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}{\Delta(t)} \geq \frac{2L-\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}{\frac{|z_i-z_j|}{v\sqrt{\frac{\kappa}{1-\kappa}}}}$$

and we already have that $(x_i-x_j)^2+(y_i-y_j)^2 < 2^2\kappa^2L^2-(z_i-z_j)^2$, so

$$\begin{aligned}\frac{2L-\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}{\Delta(t)} &> \frac{2L-\sqrt{2^2\kappa^2L^2-(z_i-z_j)^2}}{\frac{|z_i-z_j|}{v\sqrt{\frac{\kappa}{1-\kappa}}}} \\ &> \frac{(2L-\sqrt{2^2\kappa^2L^2-(z_i-z_j)^2})v\sqrt{\frac{\kappa}{1-\kappa}}}{|z_i-z_j|} \\ &> v\frac{(2L-\sqrt{2^2\kappa^2L^2-(z_i-z_j)^2})\sqrt{\frac{\kappa}{1-\kappa}}}{|z_i-z_j|}\end{aligned}$$

For sake of visual clarity, we replace $|z_i-z_j|$ with simply $|z|$.

$$> v\frac{(2L-\sqrt{2^2\kappa^2L^2-z^2})\sqrt{\frac{\kappa}{1-\kappa}}}{|z|}$$

Now, we look at $\frac{(2L-\sqrt{2^2\kappa^2L^2-z^2})\sqrt{\frac{\kappa}{1-\kappa}}}{|z|}$ in more detail.

$$\begin{aligned}\frac{\left(2L-\sqrt{2^2\kappa^2L^2-z^2}\right)\sqrt{\frac{\kappa}{1-\kappa}}}{|z|} &\overset{?}{>} 1 \\ \left(2L-\sqrt{2^2\kappa^2L^2-z^2}\right)\sqrt{\frac{\kappa}{1-\kappa}} &\overset{?}{>} |z| \\ \left(2L-\sqrt{2^2\kappa^2L^2-z^2}\right)^2\frac{\kappa}{1-\kappa} &\overset{?}{>} z^2 \\ 4L^2-4L\sqrt{4\kappa^2L^2-z^2}+4\kappa^2L^2-\left(1+\frac{1-\kappa}{\kappa}\right)z^2 &\overset{?}{>} 0 \\ 4L^2+4\kappa^2L^2-\left(1+\frac{1-\kappa}{\kappa}\right)z^2 &\overset{?}{>} 4L\sqrt{4\kappa^2L^2-z^2} \\ \left(4L^2+4\kappa^2L^2-\left(1+\frac{1-\kappa}{\kappa}\right)z^2\right)^2 &\overset{?}{>} 16L^2\left(4\kappa^2L^2-z^2\right)\end{aligned}$$

Rearranging terms, we then have

$$16L^4\left(1+\kappa^2\right)^2-64\kappa^2L^4+16L^2z^2-8L^2\left(1+\kappa^2\right)\left(1+\frac{1-\kappa}{\kappa}\right)z^2 + \left(1+\frac{1-\kappa}{\kappa}\right)^2 z^4 \overset{?}{>} 0$$

$$16L^4\left(1-2\kappa^2+\kappa^4\right)-8L^2\left(-1+\frac{1-\kappa}{\kappa}+\kappa\right)z^2+\left(1+\frac{1-\kappa}{\kappa}\right)^2 z^4 \overset{?}{>} 0$$

$$16L^4\left(1-\kappa^2\right)^2-8L^2\left(\frac{(1-\kappa)^2}{\kappa}\right)z^2+\left(\frac{1}{\kappa}\right)^2 z^4 \overset{?}{>} 0$$

We solve for z^2 in terms of L^2 and find that the roots are

$$z^2=4L^2\kappa\left(1-\kappa\right)^2\pm 8L^2\kappa\left(1-\kappa\right)\sqrt{-\kappa}.$$

The two roots are complex. In other words, the inequality is either always true or never true. We substitute the real component of the roots into the inequality.

$$16L^4\left(1-\kappa^2\right)^2-8L^2\left(\frac{(1-\kappa)^2}{\kappa}\right)\left(4L^2\kappa\left(1-\kappa\right)^2\right)+\left(\frac{1}{\kappa}\right)^2\left(4L^2\kappa\left(1-\kappa\right)^2\right)^2 \overset{?}{>} 0$$

$$16L^4\left(1-\kappa^2\right)^2-16L^4\left(1-\kappa\right)^4 \overset{?}{>} 0$$

$$16L^4\left(\left(1-\kappa^2\right)^2-\left(1-\kappa\right)^4\right) \overset{?}{>} 0$$

$$16L^4\left(4\kappa-8\kappa^2+4\kappa^3\right) \overset{?}{>} 0$$

$$64L^4\left(\kappa\right)\left(1-\kappa\right)^2 \overset{?}{>} 0$$

Recall that $0<\kappa<1$. Thus $0<\left(\kappa\right)\left(1-\kappa\right)^2\leq\frac{4}{27}$, and $64L^4\left(\kappa\right)\left(1-\kappa\right)^2>0$. The real component of the solution for z^2 is the extremum of the parabola as a

function of z^2, and the non-zero complex component means that the inequality must be always true or never true. Now we see that the inequality turns out to be true.

This shows that $\frac{\left(2L - \sqrt{2^2\kappa^2L^2 - z^2}\right)\sqrt{\frac{\kappa}{1-\kappa}}}{|z|} > 1$. In particular,

$$\frac{2L - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\Delta(t)} > v\frac{\left(2L - \sqrt{2^2\kappa^2L^2 - z^2}\right)\sqrt{\frac{\kappa}{1-\kappa}}}{|z|} > v$$

but v is the maximum speed of any point, so p_i can not be within the κL-ball of p_j.

5.2 Surface normals

In the case of the motion of adjacent points of the surface, we are looking only at local motion. We consider a point p on the surface and the motion of the local points of the surface relative to p. Because we are viewing all of the local motion relative to p, all of the motion resembles rotation about p. At its root, this behavior mimics that of a helicoid. We can thus use a helicoid centered at p to model local motion. Specifically, the surface rotates about a point along a third axis (z or t, e.g.) like a helicoid. We conjecture that a bound based on the maximum rate of change in surface normal can bound the $3d$ curvature due to local rotational motion.

5.3 Acceleration

We now show that the $3d$ local feature size due to acceleration is no worse than the $2d$ local feature size. To do this, we examine the curvature of accelerating motion and show that the radius of curvature in $3d$ due to acceleration is at least as large as the minimum local feature size in $2d$. We begin by defining the relationship between

curvature and motion.

We are given a position function $\mathbf{r}(t) = (x(t), y(t), z(t))$, with arclength

$$s = \int \sqrt{x'^2(t) + y'^2(t) + x'^2(t)}dt.$$

By definition, acceleration is $\vec{a} = \frac{d^2s}{dt^2}\hat{T} + K\left(\frac{ds}{dt}\right)^2\hat{N}$, where K is the curvature. Another way to define acceleration is $\vec{a} = \frac{d^2s}{dt^2}\hat{T} + \frac{1}{R}\left(\frac{ds}{dt}\right)^2\hat{N}$, because $K = \frac{1}{R}$, where R is the radius of curvature. $\hat{T}$ is the unit tangent vector, defined as $\frac{d\mathbf{r}}{ds}$. $\hat{N}$ is the unit normal vector, defined as $\frac{1}{K}\frac{d\hat{T}}{ds}$. In all of these, s is the arclength of the curve, a measure that is commonly used when describing curves and their properties.

In many cases, s is not practical in actual computations. A standard practice uses the chain rule to work with t instead of s. The definitions of the two unit vectors then become $\hat{T} = \frac{\frac{d\mathbf{r}}{dt}}{\frac{ds}{dt}}$ and $\hat{N} = \frac{1}{K}\frac{\frac{d\hat{T}}{dt}}{\frac{ds}{dt}}$.

The magnitude of acceleration is defined as $|\vec{a}| = \sqrt{a_T^2 + a_N^2}$ where $a_T = \frac{d^2s}{dt^2}$, the magnitude of tangent acceleration, and $a_N = K\left(\frac{ds}{dt}\right)^2$, the magnitude of normal acceleration. Because $\hat{T}$ and $\hat{N}$ are orthogonal, they relate to $|\vec{a}|$ not only as the legs of a right triangle in the Pythagorean Theorem, as in the definition of $|\vec{a}|$, but also as sine and cosine with a radius of $|\vec{a}|$. Thus, given a particular magnitude of acceleration, we know the range of possible values for each of a_T and a_N.

First, let us look more closely at $|\vec{a}|$ in both $2d$ and $3d$. A more general definition

of $|\vec{a}|$ is $|\mathbf{r}''|$. In $2d$, we have

$$\mathbf{r}(t) = (x(t), y(t))$$
$$\mathbf{r}'(t) = (x'(t), y'(t))$$
$$\mathbf{r}''(t) = (x''(t), y''(t))$$

and so

$$|\mathbf{r}''| = \sqrt{x''^2 + y''^2},$$

and in $3d$, we have

$$\mathbf{r}(t) = (x(t), y(t), z(t))$$
$$\mathbf{r}'(t) = (x'(t), y'(t), z'(t))$$
$$\mathbf{r}''(t) = (x''(t), y''(t), z''(t))$$
$$|\mathbf{r}''| = \sqrt{x''^2 + y''^2 + z''^2}$$

but $z(t) = ct$, where c is our scaling constant, so

$$\begin{aligned} |\mathbf{r}''| &= \sqrt{x'^2 + y''^2 + 0} \\ &= \sqrt{x''^2 + y''^2} \end{aligned}$$

Thus we see that $|\vec{a}|$ is the same in $2d$ and $3d$.

This tells us also that $\max_{2d} |\vec{a}| = \max_{3d} |\vec{a}|$. Thus, for both $2d$ and $3d$, our range of possible values for a_T and a_N is

$$\begin{aligned} a_T &= |\vec{a}| sin\theta \\ a_N &= |\vec{a}| cos\theta \end{aligned} \qquad \text{for } 0 \leq \theta \leq \frac{\pi}{2}.$$

Then, if we go back and review that $a_N = K\left(\frac{ds}{dt}\right)^2$, and we also see that $\left(\frac{ds}{dt}\right)$ is $\sqrt{x'^2 + y'^2}$ in $2d$ and $\sqrt{x'^2 + y'^2 + c^2}$ in $3d$, we have the following.

$$K_2 = \frac{a_N}{\left(\frac{ds}{dt}\right)^2} \qquad K_3 = \frac{a_N}{\left(\frac{ds}{dt}\right)^2}$$
$$= \frac{a_N}{\left(\sqrt{x'^2 + y'^2}\right)^2} \qquad = \frac{a_N}{\left(\sqrt{x'^2 + y'^2 + c^2}\right)^2}$$

Thus the maximum curvatures are

$$\max K_2 = \frac{\max a_N}{x'^2 + y'^2} \qquad \max K_3 = \frac{\max a_N}{x'^2 + y'^2 + c^2}$$
$$= \frac{\max |\vec{a}|}{x'^2 + y'^2} \qquad = \frac{\max |\vec{a}|}{x'^2 + y'^2 + c^2}$$

And because the denominator values are always nonnegative, we can see that $\max K_2 > \max K_3$ for all $c > 0$.

At this point, we just need to make sure that we do not break our local feature size. Going back to our analysis, we used κL as the minimum $\mathbf{R} \geq \kappa L$. Now, though, we know that the maximum $3d$ curvature can't be less than the maximum $2d$ curvature, or in other words, the minimum $2d$ radius of curvature can't be greater than the minimum $3d$ radius of curvature. Therefore, we no longer need κL. Instead, we can simply use L. Rewriting this, we have that

$$c \geq \sqrt{\kappa L f''(t)}$$

becomes

$$c \geq \sqrt{L \max |\vec{a}|}$$

If we then substitute this into our curvature equation, we have

$$\begin{aligned}
\max K_3 &= \frac{\max|\vec{a}|}{x'^2+y'^2+c^2} \\
\min \mathbf{R}_3 = \frac{1}{\max K_3} &= \frac{x'^2+y'^2+c^2}{\max|\vec{a}|} \\
&= \frac{x'^2+y'^2+c^2}{\max|\vec{a}|} \\
&\geq \frac{x'^2+y'^2+L\max|\vec{a}|}{\max|\vec{a}|} = \frac{x'^2+y'^2}{\max|\vec{a}|} + L \geq L \\
\min \mathbf{R}_3 &\geq L
\end{aligned}$$

In the case that we did want to keep κL as our minimum radius, it is easy to see that that would work, too. However, what we do see is that our original suspicion that acceleration would not pose the same restrictions as velocity was indeed correct.

We must note the special case when $\max a_N = 0$ in $2d$. This means that $\max K_2 = 0$, so long as $\frac{ds}{dt} \neq 0$, even though $\max_{2d}|\vec{a}| > 0$ may still be true. We see that $\max K_3 = \frac{\max a_N}{\left(\frac{ds}{dt}\right)^2} = \frac{\max|\vec{a}|}{\left(\frac{ds}{dt}\right)^2} > 0$, which means that $\max K_2 < \max K_3$. However, c is still calculated from $\max|\vec{a}|$, so the bound on $\max K_3$ relative to κL still holds.

Chapter 6

Conclusions

We have shown that given certain starting information (minimum $2d$ local feature size, maximum velocity, maximum acceleration and maximum change in surface normal), we can compute bounds on a scaling factor c for mapping t into the z-axis that are necessary for a good reconstruction using PowerCrust. We select κ, a constant factor to apply to L, the minimum $2d$ local feature size, and compute three bounds on the value of c using the maximum velocity, the maximum acceleration and the maximum change in surface normal, respectively. We choose c to satisfy all of these bounds. We do not at this point have a proof to show that these bounds are sufficient.

Because we used κL to determine c, we can then compute a sampling density to satisfy PowerCrust's requirement. Since $r < \frac{1}{3}$, we can sample for a density of $\frac{\kappa L}{3}$.

In the case of real-application scanning limitations, such as MRI or CT scans, the object might be moving too quickly to satisfy the scanning requirements as determined by the scaling factor. In some situations, we can make use of periodic motion to collect more scans without requiring a sudden technological leap.

While sufficient scanning may not be possible in absolute time, we can instead take scans in time modulo p, the period of the object's motion, and insert these scans into the timeframe of one period of motion. With this idea, we can take additional scans at a specific time within the period of motion but at different physical locations and also take more scans at a particular physical location over more points in the period of motion.

Chapter 7

Future Work

We have shown that the $2d + t$ problem appears to be feasible. This opens the doors to investigating the $3d + t$ problem. We can use similar mathematical analysis to consider scaling factors for t and again look at possible scanning schemes for minimizing the number of scans needed to guarantee a good $3d + t$ reconstruction. It would also be useful to implement at $4d$ surface reconstruction algorithm and analyze its accuracy and requirements. Further, we would like to find tighter bounds on c and to prove that our bounds are indeed sufficient for reconstructing a $2d$ object and its motion.

Bibliography

[1] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 213–222, New York, NY, USA, 2000. ACM.

[2] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, 1998.

[3] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, New York, NY, USA, 1998. ACM.

[4] Nina Amenta, Sunghee Choi, and Ravi Kolluri. The power crust. In *6th ACM Symposium on Solid Modeling*, pages 249–260, 2001.

[5] Nina Amenta, Sunghee Choi, and Ravi Kolluri. The power crust, union of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19(2-3):127–153, 2001.

[6] F Bernardini, I M Martin, and H Rushmeier. Highquality texture reconstruction from multiple scans. *IEEE Transactions on Visualization and Computer Graphics*, 7(4), 2001.

[7] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, Gabriel Taubin, and Senior Member. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5:349–359, 1999.

[8] Fausto Bernardini, Holly Rushmeier, Ioana M Martin, Joshua Mittleman, and Gabriel Taubin. Building a digital model of Michelangelo's Florentine Pieta. *IEEE Computer Graphics and Applications*, 22(1):59–67, 2002.

[9] H. S. M. Coxeter. *Introduction to Geometry*. John Wiley and Sons, Inc., second edition, 1989. Wiley Classics Library.

[10] H. S. M. Coxeter and S. L. Greitzer. *Geometry Revisited*. New Mathematical Library 19. Random House, 1967.

[11] Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. The Cambridge monographs on applied and computational mathematics, 23. Cambridge University Press, 2007.

[12] Tamal K. Dey and Joachim Giesen. Detecting undersampling in surface reconstruction. In *17th ACM Symposium on Computational Geometry*, pages 257–263, 2001.

[13] Tamal K. Dey and Piyush Kumar. A simple provable algorithm for curve reconstruction. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 893–894, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[14] Ludwig Eckhart. *Four-Dimensional Space*. Indiana University Press, 1968.

[15] Andriy Fedorov, Nikos Chrisochoides, Ron Kikinis, and Simon K. Warfield. Tetrahedral mesh generation for medical imaging. In *8th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2005)*, 2005.

[16] Karl Friedrich Gauss. *General Investigations of Curved Surfaces of 1827 and 1825*. The Princeton University Library, 1902.

[17] Keith Kendig. *Conics*. The Mathematical Association of America, 2005.

[18] David Koller, Jennifer Trimble, Tina Najbjerg, Natasha Gelfand, and Marc Levoy. Fragments of the city: Stanford's digital forma urbis romae project. In *Proceedings of the Third Williams Symposium on Classical Architecture*, pages 237–252, 2006.

[19] Ravikrishna Kolluri, Jonathan R. Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Symposium on Geometry Processing*, pages 11–21. ACM Press, July 2004.

[20] M Levoy, K Pulli, B Curless, S Rusinkiewicz, D Koller, L Pereira, M Ginzton, S Anderson, J Davis, J Ginsberg, J Shade, and D Fulk. The digital Michelangelo project: 3d scanning of large statues. In *Computer Graphics, SIGGRAPH 2000 Proceedings*, pages 131–144, 2000.

[21] A. Mohamed and C. Davatzikos. Finite element mesh generation and remeshing from segmented medical images. *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, 1:420–423, April 2004.

[22] Barrett O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[23] Vladimir Rovenski. *Geometry of Curves and Surfaces with MAPLE*. Birkhäuser, 2000.

[24] John W. Rutter. *Geometry of Curves*. Chapman and Hall/CRC Mathematics. Chapman and Hall / CRC, 2000.

[25] George Salmon. *A treatise on conic sections: Containing an account of some of the most important modern algebraic and geometric methods*. Longman, Brown, Green, Longman, and Roberts, 1863.

[26] Yong Shi, G. Dick van Albada, Jack Dongarra, and Peter M. A. Sloot, editors. *Computational Science - ICCS 2007, 7th International Conference Beijing, China, May 27-30, 2007, Proceedings, Part I*, volume 4487 of *Lecture Notes in Computer Science*. Springer, 2007.

[27] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*, volume Two. Publish or Perish, Inc., second edition, 1979.

[28] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*, volume Three. Publish or Perish, Inc., second edition, 1979.

[29] Dirk J. Struik. *Lectures on Classical Differential Geometry*. Addison=Wesley Series in Mathematics. Addison-Wesley Publishing Company, Inc., 1961.

[30] Dominik Szczerba, Robert H. P. McGregor, and Gábor Székely. High quality surface mesh generation for multi-physics bio-medical simulations. In Shi et al. [26], pages 906–913.

[31] G. Tabor, P. G. Young, T. Beresford West, and A. Benattayallah. Mesh construction from medical imaging for multiphysics simulation: Heat transfer and fluid flow in complex geometries. *Engineering Applications of Computational Fluid Mechanics*, 1(2):126–135, 2007.

[32] J.C.M. Teo, C.K. Chui, Z.L. Wang, S.H. Ong, C.H. Yan, S.C. Wang, H.K. Wong, and S.H. Teoh. Heterogeneous meshing and biomechanical modeling of human spine. *Medical Engineering & Physics*, 29(2):277–290, 2007.

[33] John A. Thorpe. *Elementary Topics in Differential Geometry*. Undergraduate texts in mathematics. Springer-Verlag, 1979.

[34] Victor Andreevich Toponogov. *Differential Geometry of Curves and Surfaces: A concise guide*. Birkhäuser, 2006.

[35] Wikipedia. Helicoid — wikipedia, the free encyclopedia, 2009. [Online; accessed 17-February-2009].

[36] Philippe G. Young, Gene Tabor, T. Collins, Jana Richterova, Emily Dejuniat, and Terence Beresford-West. Automating the generation of 3d finite element models based on medical imaging data. In *2006 Digital Human Modeling for Design and Engineering Conference*, 2006.