Application of Sweeping Techniques to Reverse Engineering

By
Jason Kantz

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Computer and Information Science

Department of Computer and Information Science The University of Michigan–Dearborn

December 19, 2003

Application of Sweeping Techniques to Reverse Engineering

By
Jason Kantz

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Computer and Information Science

Department of Computer and Information Science The University of Michigan–Dearborn

December 19, 2003

Approved By

_____	_____
Professor David Yoon	Date
_____	_____
Professor Jie Shen	Date
_____	_____
Professor Armen Zakarian	Date

Abstract

As 3-D scanning systems become faster and less expensive, the reverse engineering process—recreating mathematical models from physical parts—is gaining more attention. Many physical parts are created by combining a series of sweeping operations within CAD/CAM systems. By applying these sweeping techniques in the reverse, one can recreate mathematical models of many parts. This investigation of the reverse engineering process presents an approach for recovering three types of swept surfaces. Translational sweeps are found by recovering a profile curve and a trajectory curve. Rotational sweeps are found by recovering a profile curve and an axis of rotation. Free-form swept surfaces are found by slicing the data into a set of contours along an axis and skinning a surface over the section curves. These recovery techniques all depend on an approach presented for recovering curves by projecting a set of points onto a slicing plane for curve approximation. This approach to recovering the three types of swept surfaces is demonstrated on a number of examples that include data collected from parts scanned with a Minolta Vivid 900 3-D scanner.

Contents

List of Figures

Acknowledgments

I wish to express sincere appreciation to my advisor David Yoon for his help and guidance in pursuing this research. I would also like to thank committee members Jie Shen and Armen Zakarian. I wish to thank Evrard Ohou for his feedback on the implementation. Finally, I wish to thank my wife Angela for her support and encouragement as I completed this work.

Chapter 1

Introduction

1.1 Objective and Scope of Research Project

A wide variety of industries use computer aided design and manufacturing systems to engineer physical objects from mathematical models. As 3-D scanning systems become faster and less expensive, the reverse engineering process—recreating mathematical models from physical parts—is gaining more attention. Many physical parts are created by combining a series of sweeping operations within CAD/CAM systems. By applying these sweeping techniques in the reverse, one can recreate mathematical models that have certain advantages over general purpose surface fitting techniques. One advantage is that the recreated sweep model can be easily altered by changing sweep parameters. The sweep model also has the advantage that it can infer information that may be missing from the measured data. Most reverse engineering applications assume the measured parts are complete. However, there may be cases where parts are not complete due to breakage or wear. The problem addressed in this thesis is then: given 3-D point data measured from a physical swept volume or a general free-form surface that may or may not be complete, recreate the mathematical sweep components and operations to match the data. The focus will be on translational sweeps, rotational sweeps, and closed free-form surfaces.

1.2 Motivating Applications

Many industries have catalogs of parts but no electronic or paper engineering drawing for the part. When the part needs to be replaced or incorporated into new designs within a CAD system, the mathematical model of the part must be recovered. The algorithms presented in this thesis are targeted to be part of reverse engineering systems with the goal of inferring the shape of a broken or worn out part so that it may be redesigned and manufactured.

With advances in layered manufacturing [45] a complete reverse engineering system can be used to rapidly manufacture parts. Reverse engineering and rapid manufacturing are valuable in situations where parts are not readily available because the cost of transporting or keeping a new part in inventory is too high, or the part is obsolete. Spare parts for heavy construction equipment are expensive to keep on hand. If replacements for damaged parts used in machinery like cranes, graders, loaders, and excavators can be manufactured on demand, cost, risk and repair time can be greatly reduced. In remote battlefields and long-term space flights, the cost of transporting new parts is high, and can be reduced by the ability to reverse engineer and rapidly manufacture or repair broken parts.

1.3 Outline

Chapter 2 provides an overview of the reverse engineering process by studying approaches to data acquisition, preprocessing, segmentation, and surface fitting. The mathematical representation of swept surfaces is covered in chapter 3. Chapter 4 presents an approach to the recovery of 2-D curves from 3-D point clouds. Chapter 5 applies the curve recovery technique to the reconstruction of translational sweeps, rotational sweeps, and closed free-form surfaces. Chapter 6 gives a summary and suggests future work that includes an approach to segmenting a point cloud into component swept surfaces to produce a boundary representation model.

Chapter 2

Literature Survey

Váraday et al [44] introduce and survey topics in reverse engineering, breaking the process down into the following phases:

- Data Acquisition - the collection of 3D point data from scanning or use of a coordinate measuring machine (CMM)
- Preprocessing and Segmentation - organizing the point data and dividing it into subsets that form natural surfaces
- Surface Fitting - finding a surface of a given type which is the best fit to the points in a subset

2.1 Data Acquisition

Figure 2.1 gives a classification from [44] of methods for acquiring range images—sets of points in 3D. Curless surveys recent 3D scanning technologies in [11]. Contact methods are based on probes at the end of an arm and use the kinematics of the arm to determine the contact position. Optical methods are generally more popular than tactile methods because of faster acquisition rates. Optical methods are either active or passive. Passive methods use image analysis and stereo pairs of images to determine height and coordinate information. Active methods use some form of artificial light to acquire the range data. Structured lighting projects a pattern of light onto a surface and determines distance by interpreting deformations of the pattern. Ranging methods measure distance by sensing the time of flight, starting when a

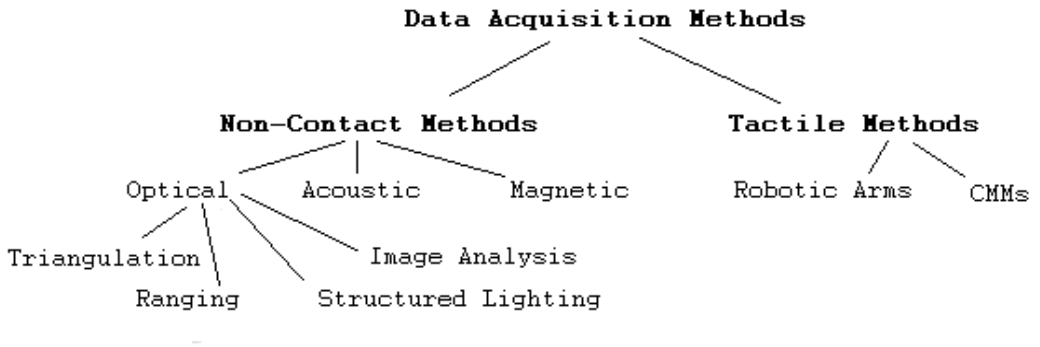

Figure 2.1: Data Acquisition Methods from [44]

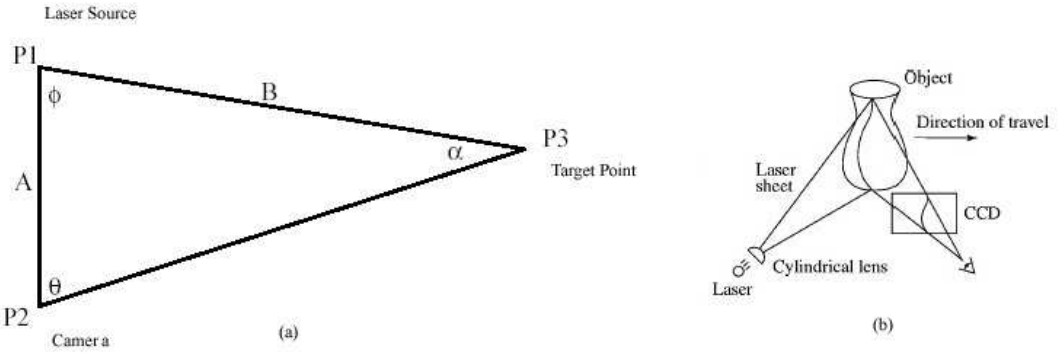

Figure 2.2: (a) 2D triangulation using a laser beam for illumination (b) extension to 3D from [11]

pulsed laser beam leaves a device and ending when it is reflected back to a sensor.

The data for this thesis was acquired using a Minolta Vivid 900 laser scanner. The Vivid 900 uses a triangulation method to acquire range data. Triangulation works by sending a focused beam of light into the scene. This beam of light creates a tiny spot of light in the scene, the target. The target is recorded by a sensor or in a Charged Coupled Device (CCD) image. The center pixel of the target is found and a line of sight can be traced to the point of reflection on the illuminated surface. Figure 2.2a shows the triangle that is formed. Given the two angles, ϕ and θ, and the distance A, the distance B can be determined based on the law of sines: $B = A\frac{sin(\theta)}{sin(\alpha)} = A\frac{sin(\theta)}{sin(\phi+\theta)}$. The Vivid 900 works by fanning the laser beam into a plane of light as shown in figure 2.2b. The scanline is passed over the surface by a rotating mirror, while the camera captures images. 3D points along the surface contours are measured by triangulating pixels in the camera's image, the line of sight, and the plane of light.

To digitize an entire object, multiple scans of the object must be taken from different angles so that every surface of the object is visible from at least one view. The process of merging of these multiple scans is called registration of the range data. There are three popular approaches to registering scans:

1. Reference spheres are attached to the object, so that at least one sphere is visible in every scan. Each data set is translated and rotated so that centers of the spheres are aligned.

2. Planar regions in overlapping scans are identified, and the data is translated and rotated to reduce the registration error. [50]

3. A rotating turn-table is calibrated to a coordinate system, then the object is placed on the turn-table and rotated and scanned from numerous angles. The scans are merged based on the turn-table's rotation from it's initial calibration.

Chen and Medioni [7] present an approach to registering multiple range images based on minimizing registration error between planes. Besl and McKay [4] present a general purpose registration technique that can be applied to point sets, line segments, triangle sets, parametric surfaces and other geometric data.

A combination of turn-table calibration and minimization of registration error between user-selected planes was used in the data acquisition for this thesis. The sides of objects were scanned and registered using the rotating turn-table, and scans of the tops and bottoms of objects were merged by selecting common points and planes among scans.

2.2 Preprocessing and Segmentation

Many algorithms for fitting smooth continuous surfaces to the range data exploit structure in the point data. A set of point data can be classified as being either random or being ordered in some way.

Sometimes data is already in a certain order according to the device or method it was collected. Lai and Ueng [19] fit surfaces of revolution to range data already organized as vertical scan lines. When data is random and unordered, preprocessing algorithms must be applied to arrange point data into a desired order and compute additional relationships. Some scanners measure and store points ordered in a regular grid which can greatly speed up necessary preprocessing. A common approach is to fit a surface to the points after they are triangulated and organized into a mesh. The triangular mesh is generated in a preprocessing step and is often chosen to be the Delaunay Triangulation of the points [9, 16]. The mesh can be used to further process the points into distinct rectangular grids that are more friendly to fitting parametric surface patches as in [18]. There are other approaches; for example, algorithms that solve the surface from contours problem [28] rely on data organized as contours lying in parallel planes. How to best divide a volume into a set of contours is known as the slicing problem. Rapid prototyping is an area where much has been written about the slicing problem [31, 34, 43]. A method for approximating a point cloud by slicing it into planar contours is presented in chapter 4.

Segmentation is a preprocessing step that divides the point data into subsets of points. Each subset of points corresponds to an individual surface that is a connected, orientable, two-dimensional manifold embedded in R^3 [14]. The subsets, when combined, make up the entire part. By dividing the point data into subsets bounded by one or more loops of edges, we are assuming a boundary representation (B-rep) model [13]. Given a point, the problem is to determine to which surface subset the point belongs. Let the measured points be given by $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The relationship between the

measured points and the original corresponding surface $\mathbf{S}_j$ is formulated as $\mathbf{x}_i = \mathbf{y}_i + \mathbf{e}_i$, where $\mathbf{y}_i \in \mathbf{S}_j$ is a point on the original surface, and $\mathbf{e}_i \in R^3$ is an error vector.

Several pieces of information are useful in the various approaches to segmentation:

1. Neighborhood information - most points that belong to a surface will be in the neighborhood of other points belonging to that surface.

2. Edge information - some points in the same neighborhood will belong to separate surfaces, and an edge curve exists that passes through the neighborhood and divides the surfaces.

3. Normal Directions and curvature - the surface normals and curvature corresponding to local surface approximations through neighborhoods can indicate edges when curvatures or the directions of the normals change dramatically within a neighborhood. Local surface approximations include triangular facets and local quadrics. Szeliski and Tonnesen [42] offer a surface fitting approach that takes advantage of a scanner that provides a normal vector for each data point.

Two common approaches to segmentation are edge detection with linking and region growing [30]. The edge-based approach to segmenting points is to search for edges between surfaces by approximating surfaces to fit local point data. At an edge between two surfaces, normals will change direction or, in the case of surfaces made tangent continuous through a blend, surface curvature will change. A sudden change in approximation normals or extrema in curvature between neighborhoods indicates an edge point between surfaces. A search for these changes can discover edge-curves within the data that are used to segment the point set into regions. After edge-curves are detected, disjoint edges must be connected or removed. Milroy et al [30] apply an edge-based approach to segment wrap-around models using Darboux frames which contain local surface normal and curvature information. The region growing approach starts with a point and expands its neighborhood into a surface by adding neighboring points with similar properties, such as normals and curvature, until there are no points that can be consistently added to that particular face. Edges are derived from the intersection of the resulting surfaces. Besl and Jain present a region growing approach to segmentation in Ref [3].

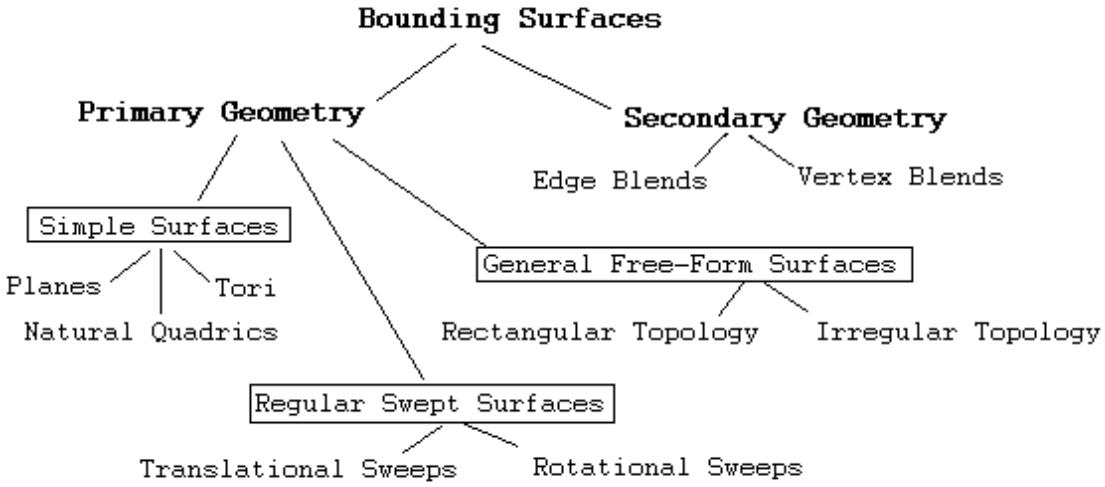

Figure 2.3: A hierarchy of surfaces [44]

Automatic segmentation for surface fitting is problematic because the reconstruction of the unknown surfaces is the motivating reason for performing segmentation in the first place. If we knew the original surface, the points could be segmented into the points near that surface. If we knew the exact subset of points that approximated a given surface, the surface could be readily recovered.

Várady et al provide a classification of possible bounding surfaces [44]. The classification reproduced in figure 2.3 indicates the possible surface types from which physical parts are often designed, and from which data points will likely be measured.

Another approach to segmentation is to rely less on automatic segmentation and rely more on user interaction to re-engineer a part. The user's goal is to identify the combination of modeling operations that will generate the part. Operations that facilitate this manual segmentation include:

1. Plane specification - user selection of three points to create a plane

2. Sectioning points by plane - being able to select or deselect the points above or below a user-specified plane

3. Curve approximation by slicing - being able to project points onto a slicing plane and approximate a curve through those points

4. Sectioning points by sweep - being able to specify a swept surface from a curve and selecting or deselecting points near that surface

5. 3D compare - this operation organizes the point cloud into a mesh and determines the distances from the scan model to the user generated surfaces, showing a color coded model where some colors correspond to high error and other colors correspond to a close fit.

With operations like these, a user can go through an iterative process of selecting subsets of points, generating surfaces, checking fit, and performing subtractions and intersections to the model to eventually recover a B-rep model for the scan.

2.3 Surface Fitting

There is a large body of literature on surface fitting. This thesis seeks to create models that consist of B-spline surfaces. Three approaches to approximating point clouds with B-spline surfaces will be considered: deformation of spheres and cylinders, approximation by serial cross-sections, and global fitting of surface patches bounded by four boundary curves.

Bae and Choi [1] present an approach to approximating surfaces for rotational free-form shapes using an orthogonal coordinate transform. First a local coordinate system is estimated. In the simplest case, this coordinate system can be located at the center of the point cloud's bounding box using principal component analysis as described in section 5.1. The point cloud is then transformed to spherical coordinates, $\{(\rho_k, \phi_k, \theta_k)\}$, relative to the estimated object center. A unit base sphere, $\mathbf{B}(u, v)$, is located with it's center at the object center. The points in the point cloud can then be thought of as the product of the base sphere and a single-valued function, $\rho_k = F(\phi_k, \theta_k)$. If $F(\phi_k, \theta_k)$ is reparameterized [6] and parameters are assigned to points in the point cloud based on their projection to the base sphere, the radii can be interpolated as a B-spline function, and the surface can be expressed as $\mathbf{S}(u, v) = F(u, v)\mathbf{B}(u, v)$. With both functions expressed in B-spline form, a symbolic product operation [38] produces the final expression for the surface. An advantage of this approach is that it can recover an exact representation

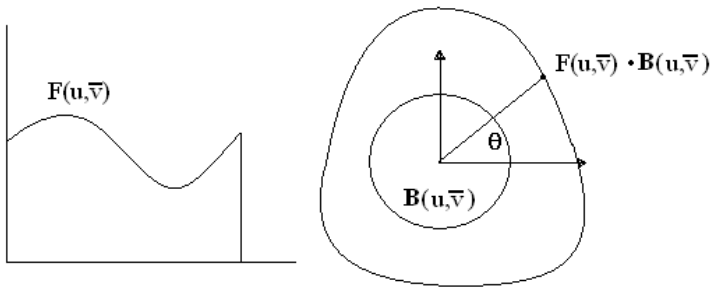

Figure 2.4: Cross-section view of the deformation of a sphere where v is held constant

when the point cloud represents a true surface of revolution. In the case of a sphere of radius R, $F(u, v) = R$, and the final representation is the base sphere scaled by a constant. The drawback of this approach is that the resulting NURBS surfaces have a high degree. When the single valued function values are interpolated with a bi-cubic, the degrees of the final surface are (5,5).

Another approach to surface fitting begins with a set of points organized as serial cross-sections. The cross-sections are approximated by NURBS curves, then skinning techniques are used to create a surface across the curves. In order to apply skinning techniques, the input curves must be compatible: they must have the same number of control points and be of the same degree. An exact approach interpolates each cross-section and uses degree-elevation and knot-insertion algorithms [36] to bring all curves to a compatible degree and number of control points. Approximation approaches are preferred because the exact approach produces an unmanageably large number of control points for the final surface.

Piegle and Tiller [37] present an algorithm that makes the curves precisely compatible, progressively removes as many knots as possible, and returns a set of curves that are compatible and that do not deviate from the precisely compatible curves by more than a chosen error bound. For loose tolerances up to 10^{-3} they found that typically 90% of the control points were unnecessary. The drawback of this approach is the large space complexity associated with the exact curve fitting.

Park and Kim [32] take a different approach, looking at the problem in

a bottom-up sense. They first apply a compact curve fitting algorithm to a single cross-section. This compact fitting algorithm uses a least squares approximation to fit the cross-section with a number of control points. The minimum number of control points required for a given tolerance is found through a binary search by performing a fit, checking the error, and either reducing or increasing the number of control points based on the error until an upper or lower bound is reached. This initiates a another binary search that seeks to find the number of control points that will fit all curves within in an error bound. A common knot vector is computed by averaging based on the first approximation curve's number of control points, and all curves are fit and tested. Depending on whether the common knot vector succeeds or fails in producing a set of curves that are all within an error bound the number of control points is decreased or increased respectively. A drawback of this approach is the tendency of least-squares fitting to break down and generate unstable solutions when the number of control points is nearly equal to the number of points in the cross-section. In a later paper [33], Park and Kim combine energy functional minimization with least-squares error minimization to overcome this instability.

Another surface fitting approach relevant to this work is the global fitting of surface patches. The surface is first segmented into patches of four boundary curves. Points within the boundary curves are sampled in a regular grid as done in Ref [18], and a constrained curve fitting procedure is applied to these points. The constraints are the boundary curves and the derivative information along the boundary curves. Refs [39, 25, 26] give methods for global and least squares approximation and parameterization of point clouds, but do not address continuity of patches. Depending on the application, the fitting procedure needs to achieve at least G^1 continuity between patches. Approaches to accomplishing this include constrained global interpolation of sampled grid points [36, p.376], local bicubic surface interpolation [36, p.399], fitting modified coons patches on to meshes [8], and setting up an optimization problem for adjusting weights and control points to stitch patches to G^1 continuity [29].

Chapter 3

Representation of Swept Volumes

Sweeping refers to the moving of a generator (which may be a curve, surface, or solid) along a trajectory curve. Blackmore et al [5] introduce the concept of a swept volume and survey swept volume research over the last twenty years while discussing applications of the work to problems in automated manufacturing. Ref [5] denotes a generator as M, a bounded closed, connected, n-dimensional submanifold of R^3 and formulates a sweep as

$$\sigma_t(x) = A(t)x + \xi(t) \tag{3.1}$$

where $x \in M$, σ is a continuous function, σ_t denotes the value of σ at t, $A : [0,1] \rightarrow SO(3)$, $\xi : [0,1] \rightarrow R^3$, and the curves $\{\sigma_t : 0 \leq t \leq 1\}$ are called the sweep trajectories of the points of M (figure 3.1).

Ref [5] focuses on reduction methods for representing swept volumes and presents a classification and formulation of swept volumes based on lie groups and differential equations. Examples of reduction methods are found in [17] and [22]. Reduction methods rely on the fact that a swept volume's boundary envelope can be constructed from the grazing points of the volume at particular t along σ_t. The grazing points are referred to as the motion silhouette in [17].

This thesis will focus on swept surfaces where the generator is a B-spline curve. Attention will be restricted to three types of sweeping techniques: translational, rotational and general. Translational sweeping moves a generator along an open trajectory curve. An example of a translational sweep

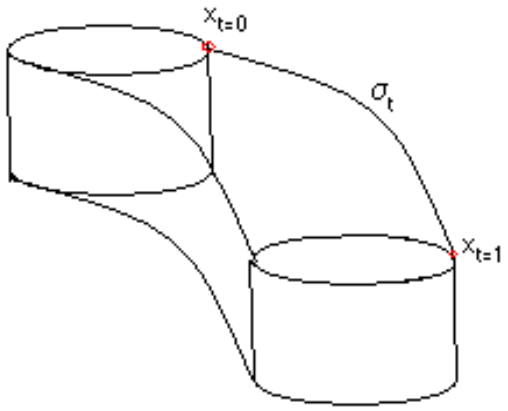

Figure 3.1: Swept Volume Formulation

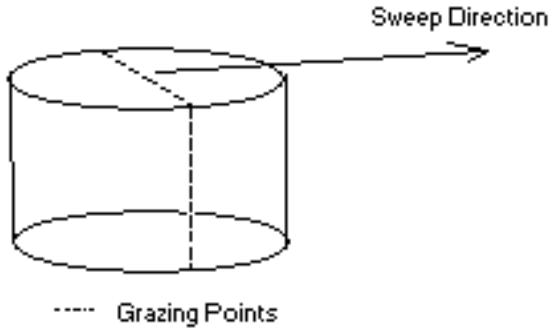

Figure 3.2: Grazing points of a swept volume

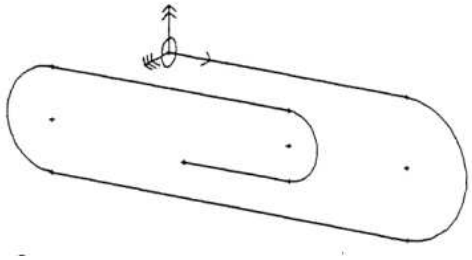

Figure 3.3: Paper clip generated with translational sweep

Figure 3.4: Rotational Sweep: a profile curve, the surface of revolution, and the shaded image

is the extrusion operation in solid modeling. Figure 3.3 depicts a translational sweep along a non-linear trajectory forming a paper clip. Rotational sweeping involves moving a profile curve along a circular trajectory giving rise to an axis of rotation. An example of a rotational sweep is the revolve operation in solid modeling. Figure 3.4 depicts a rotational sweep about the z axis forming a vase.

3.1 B-spline Curves and Surfaces

Piegl and Tiller formulate translational sweeps and rotational sweeps in terms of NURBS curves and surfaces [36]. This thesis will follow the notation in Ref [36] where a pth-degree B-spline curve is given by

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i \tag{3.2}$$

The $\{\mathbf{P}_i\}$ are the curve's control points, and the $\{N_{i,p}(u)\}$ are the pth-degree basis functions defined on the knot vector $U = \{u_i\}$. The basis functions are given by

$$U = \{\underbrace{a, \ldots, a}_{p+1}, \quad u_{p+1}, \ldots, u_{m-p-1}, \quad \underbrace{b, \ldots, b}_{p+1}\}$$

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.3}$$

Rational curves are given by

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{3.4}$$

where the $\{w_i\}$ are the weights. The notation for rational B-splines will be simplified by using homogeneous coordinates where $\mathbf{P} = H\{\mathbf{P}^w\} = H\{(wx, wy, wz, w)\} = (x, y, z)$. H is a perspective map that maps $\mathbf{P}^w$ from the origin to the hyperplane $W = 1$. Using homogeneous coordinates, rational B-splines are given as

$$\mathbf{C}^w(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i^w$$

where $H\{C^w(u)\}$ obtains points on the rational curve [36, p.29].

B-spline surfaces are denoted as a tensor product of B-spline curves.

$$\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j} \tag{3.5}$$

where the $N_{i,p}$ are pth-degree basis functions defined on a knot vector U, and the $N_{j,q}$ are qth-degree basis functions defined on a knot vector V. The $\{\mathbf{P}_{i,j}\}$ form a bidirectional net of control points for the surface.

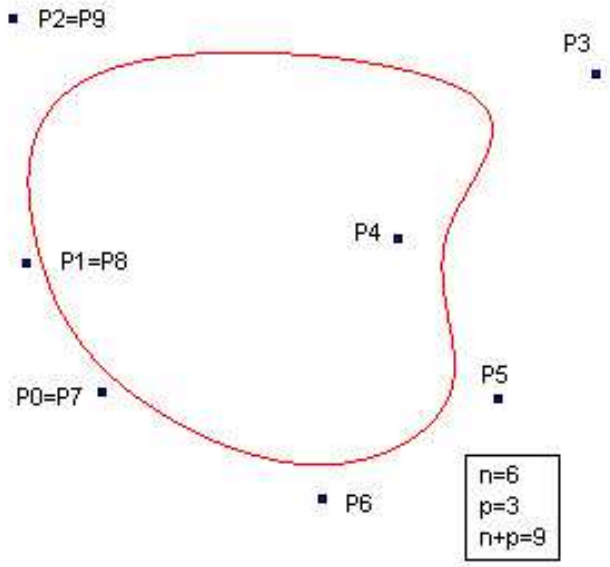

Figure 3.5: Closed Bspline Curve

Like the rational curves, rational B-spline surfaces are conveniently represented with homogeneous coordinates.

$$\mathbf{S}^w(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}^w_{i,j}$$

where $H\{S^w(u,v)\}$ obtains points on the rational surface.

3.2 Closed B-spline Curves

A pth degree closed contour curve is defined by:

$$C(u) = \sum_{i=0}^{n+p} N_{i,p}(u)P_{i_{mod(n+1)}} \qquad (0 \le u \le 1) \tag{3.6}$$

where $n+1$ is the number of distinct control points. The knot vector U of the closed contour is given by:

$$\{u_0, \ldots, u_{p-1}, \underbrace{u_p, \ldots, u_{p+n+1},} u_{p+n+2}, \ldots, u_{2p+n+1}\}$$

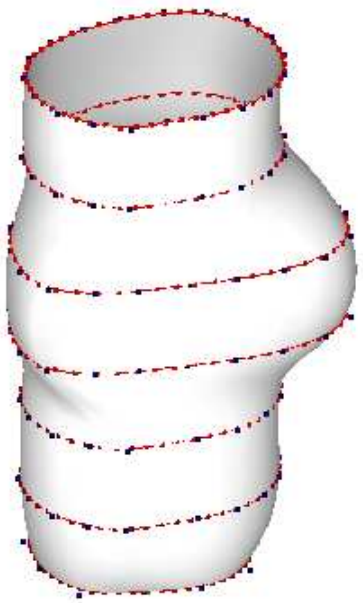

Figure 3.6: B-spline surface closed in u direction

where knots $u_p \dots u_{p+n+1}$ are known as the domain knots [32]. The first and last $p-1$ knots depend on the domain knots and are given as:

$$u_k = u_{k+1} + u_{n+k+1} - u_{n+k+2} \qquad (0 \leq k \leq p-1) \tag{3.7}$$

$$u_k = u_{k-1} + u_{k-n-1} - u_{k-n-2} \qquad (p+n+2 \leq k \leq 2p+n+1) \tag{3.8}$$

3.2.1 Example of closed B-spline curve

The closed, cubic B-spline contour with $n+1 = 7$ control points in figure 3.5 is defined by:

$$C(u) = \sum_{i=0}^{9} N_{i,3}(u) P_{i_{mod(7)}} \qquad (0 \leq u \leq 1)$$

The uniform, unclamped knot vector is

$$\{-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

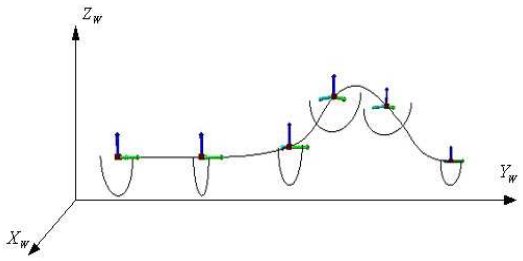

Figure 3.7: A set of section curves and their local frames from [10]

3.3 Closed B-spline surfaces

A B-spline surface closed in the u direction and open in the v direction is defined as the tensor product of B-spline curves:

$$S(u,v) = \sum_{j=0}^{m} \sum_{i=0}^{n+p} N_{i,p}(u) N_{j,q}(v) P_{i_{mod(n+1)},j} \tag{3.9}$$

where the control net consists of $(n+1) \times (m+1)$ distinct points. Figure 3.6 shows a B-spline surface closed in the u direction.

3.4 Translational Sweeps

A translational sweep is given by

$$S(u,v) = T(v) + M(v)C(u) \tag{3.10}$$

The generator curve, $C(u)$, is restricted to being a closed B-spline curve in this thesis. The trajectory curve, $T(v)$, is represented as an open B-spline curve and restricted to planar curves. $M(v)$ is a 3×3 matrix. $M(v)$ scales and rotates $C(u)$ as a function of v. When $M(v) = I$ and the trajectory is linear, the resulting surface is a general cylinder. When $M(v)$ changes with v, $M(\bar{v}_k)$ denotes the orientation of a local frame for $C(u)$ along the trajectory. Figure 3.7 depicts several local frames for $C(u)$ along a trajectory.

When $M(v) = I$, the B-spline representations of a generalized translational sweep are

$$T^w(v) = \sum_{j=0}^{m} N_{j,q}(v) T_j^{w_j}$$

$$C^w(u) = \sum_{i=0}^{n} N_{i,p}(u) Q_i^{w_i}$$

$$S^w(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j}^{w_{i,j}} \tag{3.11}$$

where $P_{i,j} = T_j + Q_i$ by the affine invariance property of B-spline curves, knot vectors V and U are those for $T(v)$ and $C(u)$, and the weight $w_{i,j} = w_i w_j$.

When $M(v) \neq I$, $M(v)$ cannot generally be represented in B-spline form. In this case $S(u,v)$ can be represented as a B-spline approximation by lofting a surface across a set of section curves $\{S(u, \bar{v}_k)\}$ at local frames along $T(v)$.

3.5 Rotational Sweeps

3.5.1 Constructing NURBS Circles

Detailed discussion of NURBS circles can be found in [35]. The construction of NURBS Circles can be broken down into two problems:

1. Constructing circular arcs less than 180 °
2. Piecing together the arcs into a full circular curve

A circular arc that is less than 180 ° can be constructed as a second degree ($p = 2$) NURBS curve from three control points, where the three control points form an isosceles triangle. Let θ be the *base angle*—the pair of equal angles in the isosceles triangle formed by the control points. The angle of the circular arc will be $2\,\theta$. The weight of the point that is not part of the base of the triangle is set to $cos(\theta)$. The weights of the control points along the base of the triangle are set to 1. A 90 ° arc has the form

$$C(u) = \sum_{i=0}^{2} N_{i,2}(u) P_i \tag{3.12}$$

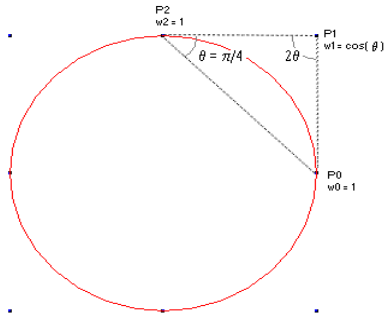

Figure 3.8: Nine control points forming a NURBS circle

where $\angle P_1P_2P_0 = \frac{\pi}{4}$, $\angle P_1P_0P_2 = \frac{\pi}{4}$, $U = \{0,0,0,1,1,1\}$ and $w_i = \{1, \frac{\sqrt{2}}{2}, 1\}$. The arcs can be pieced together using double knots to give a NURBS circle with a nine-point square control polygon:

$\{P_i\} = \{(1,0),(1,1),(0,1),(-1,1),$
$(-1,0),(-1,-1),(0,-1),(1,-1),(1,0)\}$

$$U = \{0,0,0,\frac{1}{4},\frac{1}{4},\frac{1}{2},\frac{1}{2},\frac{3}{4},\frac{3}{4},1,1,1\}$$

$$\{w_i\} = \{1,\frac{\sqrt{2}}{2},1,\frac{\sqrt{2}}{2},1,\frac{\sqrt{2}}{2},1,\frac{\sqrt{2}}{2},1\}$$

$$C^w(u) = \sum_{i=0}^{8} N_{i,2}(u)P_i^w$$

3.5.2 Revolving the Generatix

The curve, $C(v)$, revolved around an axis to create a surface of revolution is called the generatix [36, 340]. A surface of revolution, $S(u,v)$, has particular isoparametric curves at fixed $\bar{u}$ and fixed $\bar{v}$:

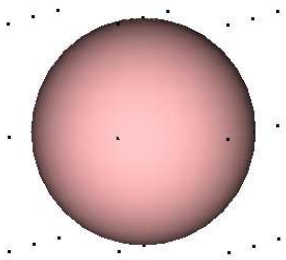

Figure 3.9: NURBS sphere generated by rotating a NURBS circle

- $S(\bar{u}, v)$ is the generatix rotated some angle about the axis of rotation.
- $S(u, \bar{v})$ is a circle in the plane perpendicular to the axis of rotation, with its center on the axis.

Constructing a surface of revolution from the B-spline circle with a nine-point control polygon gives an equation of the following form:

$$S(u, v) = \sum_{i=0}^{8} \sum_{j=0}^{m} R_{i,2;j,q}(u, v) P_{i,j} \tag{3.13}$$

where the control points $P_{0,0}$ through $P_{0,m}$ make up the control points for the generatix. The control net for the surface of revolution is set up by taking each control point of the generatix and placing it along the nine points of the square control polygon for a circle in the plane perpendicular to the axis of rotation with its center at the axis of rotation, so that $P_{0,j} = P_{i,j}$ for each j along the generatix.

The U knot vector is the knot vector of the nine point circle, and the V knot vector is the knot vector of the generatix. The weight $w_{i,j}$ is taken as the product of the circle weight, w_i, and the generatix weight, w_j.

Chapter 4

Slicing

4.1 Slicing a 3-D point cloud to recover a 2-D curve

When a 3-D point cloud is measured from a swept surface, the surface can be recovered by finding appropriate planar curves and recreating the sweep. In the case of a translational sweep, the surface depends on a profile curve and a trajectory curve. A rotational sweep depends on an axis and a profile curve. A free-form surface depends on a set of section curves that approximate the surface.

Each curve can be recovered by examining the points near an appropriate slicing plane. The slicing plane approximates the position and orientation of an original plane containing the desired curve. Methods for choosing a plane

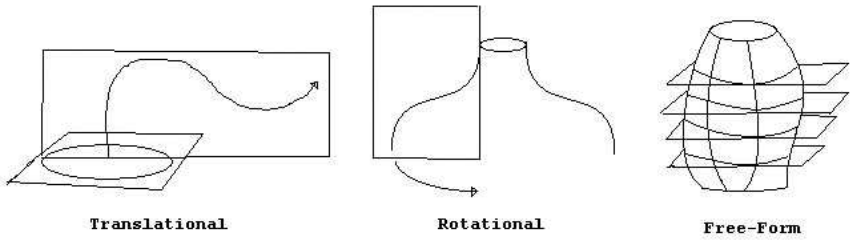

Figure 4.1: Planar curves within swept surfaces.

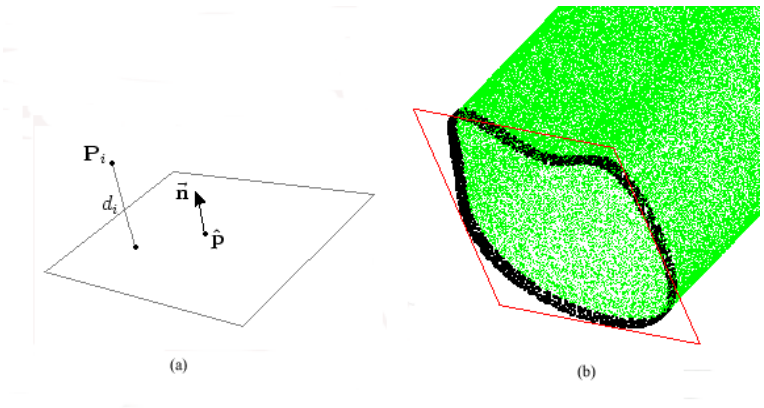

Figure 4.2: (a) Representation of the plane (b) Points near plane

according to the type of sweep (extrusion, rotation, free-form) are discussed in chapter 5. Here the concern is focused on finding a 2-D profile curve that approximates the points near a slicing plane through the point cloud. The point cloud is given as a set of points relative to the world coordinate system of the scanning device. The plane is represented in point-normal form, where the unit normal to the plane is given by $\vec{\mathbf{n}}$ and a point on the plane is given by $\hat{\mathbf{p}}$.

The curve that lies in the slicing plane can be found by projecting points near the plane onto the plane and approximating a curve to fit the projected points.

The signed distance, d_i, from some point $\mathbf{P}_i$ to the plane is given by

$$d_i = \langle \vec{\mathbf{n}}, \mathbf{P}_i \rangle + d \tag{4.1}$$

where $d = \langle -\vec{\mathbf{n}}, \hat{\mathbf{p}} \rangle$. For a derivation of this formula see Ref[41].

When t is a chosen tolerance, the points meeting the criteria $|d_i| \leq t$, translated to the plane, form a new set of points $\{\mathbf{P}'_i\}$.

$$\mathbf{P}'_i = \mathbf{P}_i - d_i \vec{\mathbf{n}} \tag{4.2}$$

The 2-D point cloud forms an unorganized band of points around the desired curve as shown in figure 4.3. The next step is to select a subset of the points as shown in figure 4.5 such that if the points are connected by line segments the shape of the curve is accurately approximated. This subset of points can then be interpolated using a B-spline curve.

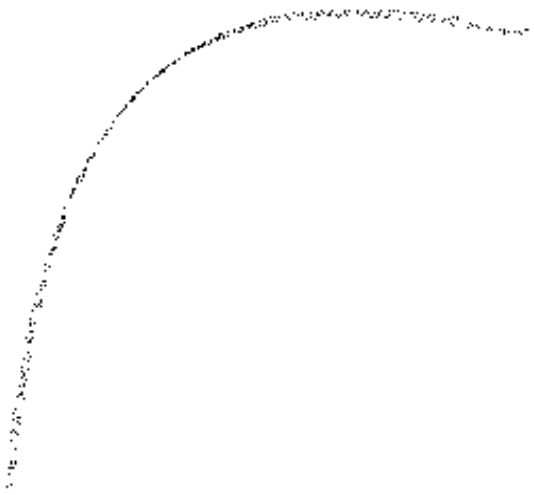

Figure 4.3: Close-up view of the band of 2-D points.

Lee [20] shows that a subset of points for curve approximation can be selected by eliminating neighboring points within a radius. Let the neighboring points that are within some radius r of a random starting point $\mathbf{P}_*$ be given as the set B. The points in B are nearly linear if the width of the 2-D point cloud is sufficiently thin. B can be broken into two sets by computing a regression line through the points in B with direction vector $\mathbf{L}_*$. The two sets are based on the angles between each vector $\vec{v}_j$ from $\mathbf{P}_*$ to $\mathbf{P}_j$ and the regression line $\mathbf{L}_*$:

$$B_1 = \{\mathbf{P}_j | \mathbf{P}_j \in B, \langle \vec{v}, \mathbf{L}_* \rangle \geq 0\}$$

$$B_2 = \{\mathbf{P}_j | \mathbf{P}_j \in B, \langle \vec{v}, \mathbf{L}_* \rangle < 0\}$$

where $\langle\rangle$ is the dot product operation. The two sets are essentially the two sides of a line through $\mathbf{P}_*$ perpendicular to $\mathbf{L}_*$.

Two points F_1 and F_2 that are points furthest from $\mathbf{P}_*$ in sets B_1 and B_2 respectively are retained while the other points in B are dropped as shown in figure 4.4. The process continues as either F_1 or F_2 becomes the next $\mathbf{P}_*$ until all the points have been considered, resulting in a thinned set of points as in figure 4.5.

This method of thinning out the data presents some potential problems.

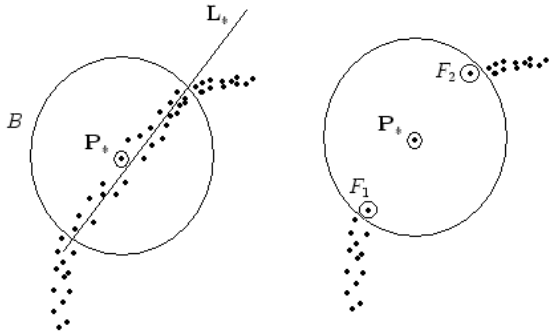

Figure 4.4: Eliminating neighboring points within a radius.

1. The time complexity of a search for the neighbors of each point could potentially have a time complexity of $O(n^2)$.

2. The point cloud may not be sufficiently thin resulting in an approximation that contains wiggles.

3. A large radius for the neighboring points can result in the exclusion of some feature points.

The following sections address these problems. The plane projected points are transformed to 2-D coordinates and a binning data structure and algorithm are applied to find neighbors in linear time, and moving least squares approximations with an adaptive radius are used to thin the point cloud.

4.2 Converting projected points to 2-D coordinates

Once the points have been translated to the plane, their coordinates are still three-dimensional. To transform the 3-D planar curve fitting problem into a 2-D curve fitting problem, the set of points $\{\mathbf{P}'_i\}$ is translated and rotated so that the normal to the plane coincides with the z axis of the world coordinate system.

First the plane is moved to the origin of the world coordinate system. The shortest vector $\vec{\mathbf{v}}$ from the plane to the origin of the world coordinate

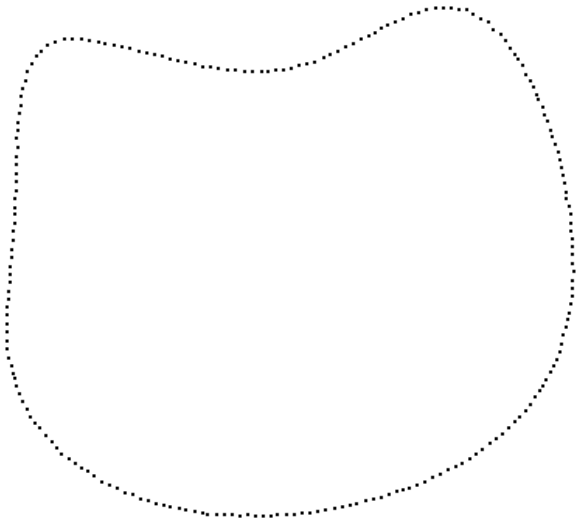

Figure 4.5: Subset of 2-D point cloud.

system can be determined from equation (4.1). The translation applied to each point is

$$\mathbf{P}'_i = \mathbf{P}'_i - \vec{\mathbf{v}} \tag{4.3}$$

where $\vec{\mathbf{v}} = d\vec{\mathbf{n}}$ and the signed distance to the origin is $d = \langle -\vec{\mathbf{n}}, \hat{\mathbf{p}} \rangle$.

Next the plane normal, $\vec{\mathbf{n}}$, is rotated about the x axis of the world coordinate system to the xz plane as shown in figure 4.6. $\vec{\mathbf{n}}$ is a unit vector originating from the origin. Rotating $\vec{\mathbf{n}}$ about the x axis by α radians will put $\vec{\mathbf{n}}$ in the xz plane. Projecting $\vec{\mathbf{n}}$ onto the yz plane gives a vector of length $h = \sqrt{\vec{n}_y^2 + \vec{n}_z^2}$.

The rotation that puts $\vec{\mathbf{n}}$ in the xz plane is given by

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{\vec{n}_z}{h} & \frac{-\vec{n}_y}{h} \\ 0 & \frac{\vec{n}_y}{h} & \frac{\vec{n}_z}{h} \end{bmatrix} \tag{4.4}$$

Finally, as shown in figure 4.7, the plane normal, $\vec{\mathbf{n}}$, is rotated about y axis to coincide with z axis. The rotation $\mathbf{R}_y$ is given by

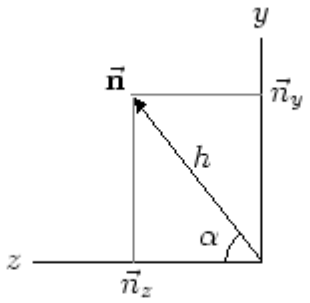

Figure 4.6: A rotation about x axis of the world coordinate system by α.

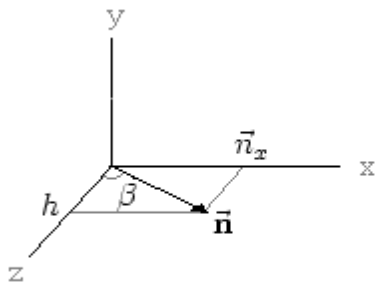

Figure 4.7: A rotation about the y axis of world coordinate system by β.

$$
\begin{aligned}
\mathbf{R}_y &= \begin{bmatrix} \cos(-\beta) & 0 & -\sin(-\beta) \\ 0 & 1 & 0 \\ \sin(-\beta) & 0 & \cos(-\beta) \end{bmatrix} \\
&= \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \\
&= \begin{bmatrix} \frac{h}{r} & 0 & \frac{\vec{n}_x}{r} \\ 0 & 1 & 0 \\ \frac{-\vec{n}_x}{r} & 0 & \frac{h}{r} \end{bmatrix}
\end{aligned} \tag{4.5}
$$

where $r = \sqrt{\vec{n}_x^2 + h^2}$.

4.3 Finding nearest neighbors within a 2-D point cloud in linear time

Given a 2-D point cloud $\{\mathbf{P}_i\}$, $0 \leq i \leq n$, and a point $\mathbf{P}_*$ there is a set of points satisfying $||\mathbf{P}_i - \mathbf{P}_*|| < r$, where r is some chosen radius. In order to find the set of points that satisfy this criteria, a comparison of distances from each point to every other point should be avoided, as this operation would have an $O(n^2)$ time complexity. The time complexity of the problem can be reduced by taking a trade off in space complexity and setting up an appropriate data structure.

Finding nearest neighbors is a well-known problem in computational geometry, and methods have been developed that use preprocessing techniques to produce the Voronoi diagram of the point set [16]. Piegl and Tiller [40] present an algorithm that avoids sophisticated preprocessing and runs in linear time using a grid based data structure. Their algorithm finds k nearest neighbors. With a slight modification, the same approach will find neighboring points within a chosen radius.

The grid data structure is set up over the min-max box $[xl, xr] \times [yb, yt]$ of the point cloud. The size of the grid is estimated as

$$
size = \alpha \sqrt{\frac{(xr - xl)(yt - yb)}{n}}
$$

where $\alpha = 1.0$ and can be changed to adjust the grid size.

The resolution of the grid in the x and y directions is given as

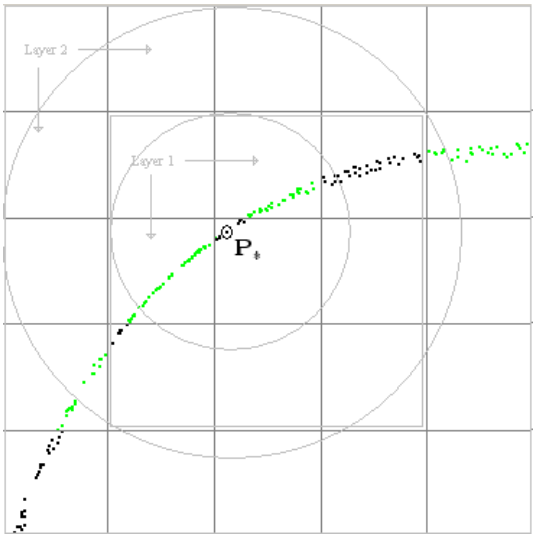

Figure 4.8: Grid over points and maximum searchable radius within layers.

$$xres = \left\lfloor \frac{xr - xl}{size} \right\rfloor \qquad yres = \left\lfloor \frac{yt - yb}{size} \right\rfloor$$

A cell structure in row-major order is created:

$$cell[j][k] \qquad j = 0, \ldots, yres - 1 \qquad k = 0, \ldots, xres - 1$$

The index of each point in $\{\mathbf{P}_i\}$ is placed in a cell. Each cell can be set up as a linked list or an adjustable vector. Indexes of points falling in the same cell are pushed onto the end of the list or vector. Given an x value or a y value the appropriate cell is found as

$$cell_j(y) = \begin{cases} yres - 1 & \text{if } y = yb \\ \left\lfloor \frac{yt - y}{size} \right\rfloor & \text{otherwise} \end{cases}$$

$$cell_k(x) = \begin{cases} xres - 1 & \text{if } x = xr \\ \left\lfloor \frac{x - xl}{size} \right\rfloor & \text{otherwise} \end{cases}$$

The search for neighboring points within a radius proceeds by beginning with the cell containing $\mathbf{P}_*$. The distances from $\mathbf{P}_*$ to each point within the

same cell are checked. The maximum radius searched within the cell is the shortest distance from $\mathbf{P}_*$ to a cell wall denoted as d_{sh}. If d_{sh} is less than the chosen radius, the search proceeds to the surrounding cells, $layer = 1$ shown in figure 4.8. The search proceeds layer by layer until a layer is reached where $d_{sh} + layer * size > r$.

4.4 Making the 2-D point cloud sufficiently thin

Levin [23] uses a method called moving least-squares to thin a point cloud. The moving least squares method was developed by McLain [27]. Lee [20] presents algorithms for applying moving least squares to point clouds with varying thickness.

The idea of moving least-squares is that point clouds can be thinned and smoothed by projecting each point onto its local regression curve. For each point, $\mathbf{P}_*$ in $\{\mathbf{P}_i\}$, there is a local quadratic regression curve, $\mathbf{Q}_* : y = a_0 + a_1 x + a_2 x^2$.

The local regression curve can be computed by minimizing:

$$d_q = \sum_{i=1}^{N} (y_i - (a_0 + a_1 x_i + a_2 x_i^2))^2 w_i$$

where d_q is the squared distance in the y direction from a neighboring point $\mathbf{P}_i = (x_i, y_i)$ to the local regression curve, and where w_i is a nonnegative weight for each $\mathbf{P}_i$. To make the approximation local to $\mathbf{P}_*$, a weighting function is chosen that penalizes points whose distance from $\mathbf{P}_*$ is larger than some chosen r:

$$wi = \begin{cases} 2\frac{h^3}{r^3} - 3\frac{h^2}{r^2} + 1 & \text{if } h \leq r, \\ 0 & \text{if } h > r \end{cases}$$

where $h = ||\mathbf{P}_i - \mathbf{P}_*||^2$.

The projection of a point onto its local regression curve can be simplified by computing a local regression line $\mathbf{L}_* : y = a_0 + a_1 x$, where a_0 and a_1 are values that minimize the squared y-distance to $\mathbf{L}_*$ from points in $\{\mathbf{P}_i\}$ near $\mathbf{P}_*$.

Next, a transformation, M, is applied to all of the points in $\{\mathbf{P}_i\}$. M is a transformation that moves $\mathbf{P}_*$ to the origin and rotates $\mathbf{L}_*$ so that it

is parallel to the x axis. The application of M produces a new set $\{\hat{\mathbf{P}}_i = (\hat{x}_i, \hat{y}_i) \,|\, i = 1, \ldots, N\}$. A quadratic regression curve $\hat{\mathbf{Q}}_*$ is computed for $\{\hat{\mathbf{P}}_i\}$ by minimizing:

$$d_q = \sum_{i=1}^{N} (\hat{y}_i - (a_0 + a_1 \hat{x}_i + a_2 \hat{x}_i{}^2))^2 w_i$$

Since $\mathbf{L}_*$ has been rotated parallel to the x axis, the projection of $\hat{\mathbf{P}}_*$ onto $\hat{\mathbf{Q}}_*$ is simply $(0, c)$, and $\mathbf{P}_*$ projection onto $\mathbf{Q}_*$ is found by applying M^{-1} to $(0, c)$.

When each point in $\{\mathbf{P}_i\}$ is projected onto its local regression curve in this manner, the point cloud is made sufficiently thin to select a subset of points for curve approximation as explained in section 4.1.

4.5 A slicing algorithm

To produce a curve from projections of points near a planar slice through a point cloud, the problem is first transformed into a 2-D problem (section 4.2). Then a grid structure is set up over the set of 2-D points (section 4.3) to perform the point thinning (section 4.4) and subset selection (section 4.1). The process of thinning the points in preparation for the ordered selection of a subset of points is implemented as follows:

Thin Point Cloud

For each point $\mathbf{P}_*$ in point cloud

Choose RADIUS for neighborhood

Enlarge RADIUS by RADIUS_INCREMENT until correlation of points is nearly linear or until a maximum number of iterations is reached or until a minimum number of neighbors is reached

approximate neighborhood of points within RADIUS of $\mathbf{P}_*$ with a quadratic curve $\mathbf{Q}_*$

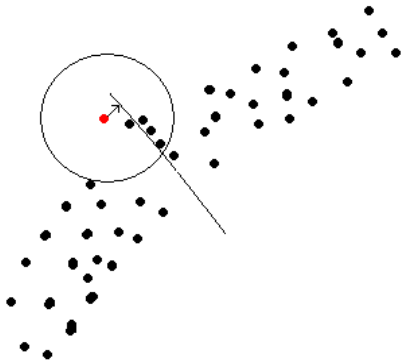

Figure 4.9: A moving least squares radius that is too small may not move the current point toward the rest of the data.

project $\mathbf{P}*$ onto $\mathbf{Q}_*$

repeat approximation and projection until average approximation error is $<$ tolerance or until a maximum number of allowed iterations

An important step of the thinning process is choosing an appropriately sized radius for collecting neighboring points of $\mathbf{P}_*$. Lee [20] presents a method for choosing an adaptive radius based on measuring the correlation. When the starting radius is too small, neighborhoods with strong correlation may be chosen that do not necessarily reflect the larger trend of the data (see figure 4.9). If the starting radius is too large, it may include unwanted points.

A starting radius that facilitates the collection of neighboring points is one that is some fraction of the size of a cell in the grid structure (equation 4.3). The points in the first layer around the cell are then collected and tested for correlation. Points can be added to the neighborhood by layer until the correlation shows an acceptable level of linearity. This approach then depends on choosing an appropriate α for equation 4.3 to create a large enough cell size to capture the width of the point cloud.

Chapter 5

Recovery of Swept Surfaces

To recover a swept surface from a 3-D point cloud, one must find the appropriate planar curves and recreate the sweep operation. In the case of a translational sweep, the surface depends on a profile curve and a trajectory curve. A rotational sweep depends on an axis and a profile curve. A free-form surface depends on a set of section curves that approximate the surface. In some cases a good estimate of a plane containing the appropriate curve can be derived from the bounding box of the set of points. The next section explains a method for finding the object center and bounding box of a 3-D point cloud. Then, in the sections that follow, the method for finding a bounding box is applied in the process of curve and surface recovery from parts represented as translational, rotational, and free-form sweeps.

5.1 Finding the object center for a set of points

The goal is to find a transformation that maps the points from the world coordinate system of the scanning device to a basis where the points are distributed evenly about each axis of a new coordinate system. The location of this object coordinate system is the center of the scanned object. This transformation can be found using the statistical method principal component analysis [21]. The axis along which the data vary the most is the primary principal component, and the minimum and maximum values along these axes form the smallest bounding box of the object.

The distribution of the points about any two axes can be measured in terms of their covariance. Covariance is a measure of how much the deviations

of two sets match. Given two sets $\{x_i\}$ and $\{y_i\}$, the covariance is

$$\sum_{i=0}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

where $\bar{x}$ and $\bar{y}$ are averages. If the points are evenly distributed about all three axes, there will be no correlation between any two axes, and the covariance between points along any two axes will be zero.

To measure the distribution of the point cloud about all three axes, first the average position $\bar{\mathbf{P}}$ of the n points is calculated.

$$\bar{\mathbf{P}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{P}_i \tag{5.1}$$

Then a covariance matrix is computed based on the average position:

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{P}_i - \bar{\mathbf{P}})(\mathbf{P}_i - \bar{\mathbf{P}})^\top$$

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}\begin{bmatrix} (x_i - \bar{x})^2 & (xi - \bar{x})(y_i - \bar{y}) & (xi - \bar{x})(z_i - \bar{z}) \\ (xi - \bar{x})(y_i - \bar{y}) & (y_i - \bar{y})^2 & (yi - \bar{y})(z_i - \bar{z}) \\ (xi - \bar{x})(z_i - \bar{z}) & (yi - \bar{y})(z_i - \bar{z}) & (z_i - \bar{z})^2 \end{bmatrix} \tag{5.2}$$

The covariance matrix is a real symmetric matrix. The non-diagonal entries correspond to covariances among the different pairings of axes. When the matrix is diagonal there is no correlation between points relative to any two axes, meaning the points are evenly distributed about all axes.

To find these axes for a given point cloud, assume a transformation matrix $\mathbf{A}$ that represents the desired transformation. $\mathbf{A}$ is a matrix that transforms each point from the world coordinate system to an object coordinate system where the points are distributed equally about all axes. This means that the covariance matrix of the transformed set of points, $\mathbf{C}'$, must be a diagonal matrix.

$$\mathbf{C}' = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{A}\mathbf{P}_i - \mathbf{A}\bar{\mathbf{P}})(\mathbf{A}\mathbf{P}_i - \mathbf{A}\bar{\mathbf{P}})^\top \tag{5.3}$$

Substituting the covariance matrix of $\{\mathbf{P}_i\}$ into equation (5.3) gives,

$$\mathbf{C}' = \mathbf{A}\mathbf{C}\mathbf{A}^\top$$

which means $\mathbf{A}^\top$ must be a matrix that diagonalizes the covariance matrix $\mathbf{C}$. A method for finding such a matrix relies on two premises:

1. The eigenvectors of a symmetric matrix are orthogonal.
2. Given a matrix $\mathbf{A}^\top = [\mathbf{V}_1\mathbf{V}_2 \dots \mathbf{V}_n]$ where the $\mathbf{V}_i$ are the eigenvectors of a matrix $\mathbf{C}$, $\mathbf{C}\mathbf{A}^\top$ produces a diagonal matrix.

The eigenvectors of a $n \times n$ matrix $\mathbf{C}$ are the n non-zero vectors V_i that are only changed in magnitude when transformed by $\mathbf{C}$:

$$\mathbf{C}\mathbf{V}_i = \lambda_i \mathbf{V}_i \tag{5.4}$$

λ_i are eigenvalues of $\mathbf{C}$, and $\mathbf{V}_i$ are eigenvectors of $\mathbf{C}$.
Equation (5.4) can be rewritten as

$$\mathbf{C}\mathbf{V}_i - \lambda_i \mathbf{V}_i = 0$$

$$(\mathbf{C} - \lambda_i \mathbf{I})\mathbf{V}_i = 0 \tag{5.5}$$

where $\mathbf{I}$ is an identity matrix. Since $\mathbf{V}_i$ are non-zero vectors, $(\mathbf{C} - \lambda_i \mathbf{I})$ is singular and not invertible. Since the determinant of a singular matrix is zero, the eigenvalues can be found as the roots of

$$\det(\mathbf{C} - \lambda_i \mathbf{I}) = 0 \tag{5.6}$$

which is an n degree polynomial in λ called the characteristic polynomial of the matrix $\mathbf{C}$. Then for each eigenvalue, an eigenvector can be found by solving the linear system in (5.5). The three eigenvectors, $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$, of the 3×3 covariance matrix $\mathbf{C}$, are the three orthogonal vectors forming natural axes for the point cloud. The vector corresponding to the largest λ_i, is the principal axis along which the data vary the most.

Once the vectors are known, they are given a position at the center of the point cloud's bounding box along the direction of the vectors and can be used as the object coordinate system. This bounding box can be found as the minimum and maximum projections of the points onto the eigenvectors:

$$\min\langle \mathbf{V}_i, \mathbf{P}_i \rangle \qquad \max\langle \mathbf{V}_i, \mathbf{P}_i \rangle$$

where $\langle\rangle$ is the dot product operation.

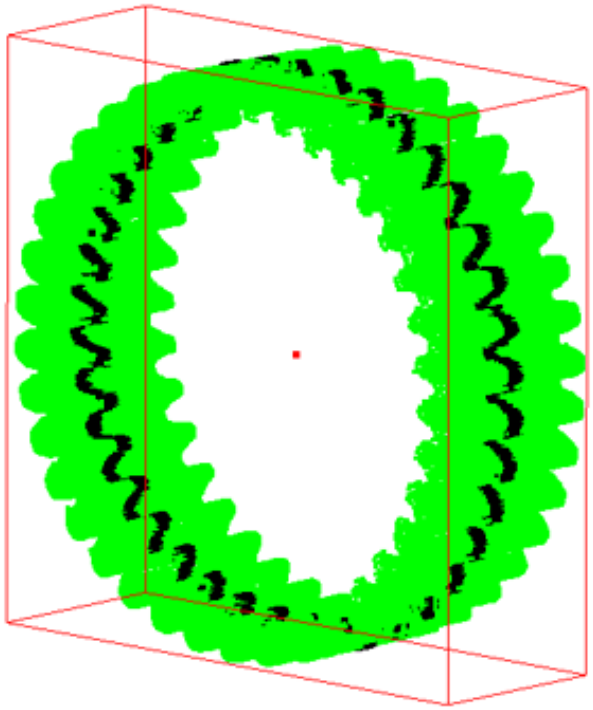

Figure 5.1: Gear point cloud with object center, bounding box, and selected plane for recovering profile curve.

5.2 Translational Sweep Recovery

5.2.1 Linear Sweep Trajectory

The first test case is a steel gear. The point cloud of the gear comes from a scan performed by the Minolta Vivid 900 where multiple scans were registered using the Raindrop Geomagic software that accompanied the system.

This implementation relies on the operator's judgment to determine if the part is suitable for a translational sweep. In the case of a linear translational sweep, four planes of the bounding box will be tangent to sides of the cylinder, and two opposite planes will contain the profile of the sweep.

The operator selects the slicing plane (figure 5.1) for the steal gear as one of the planes of the object coordinate system determined as described in section 5.1.

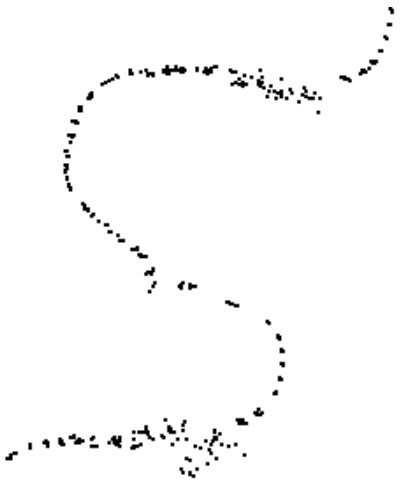

Figure 5.2: 2-D Close-up of slice of gear profile

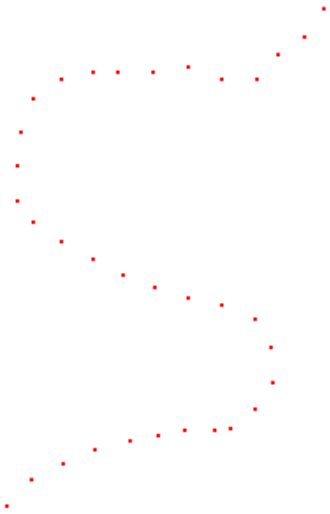

Figure 5.3: Thin points from slice of gear profile

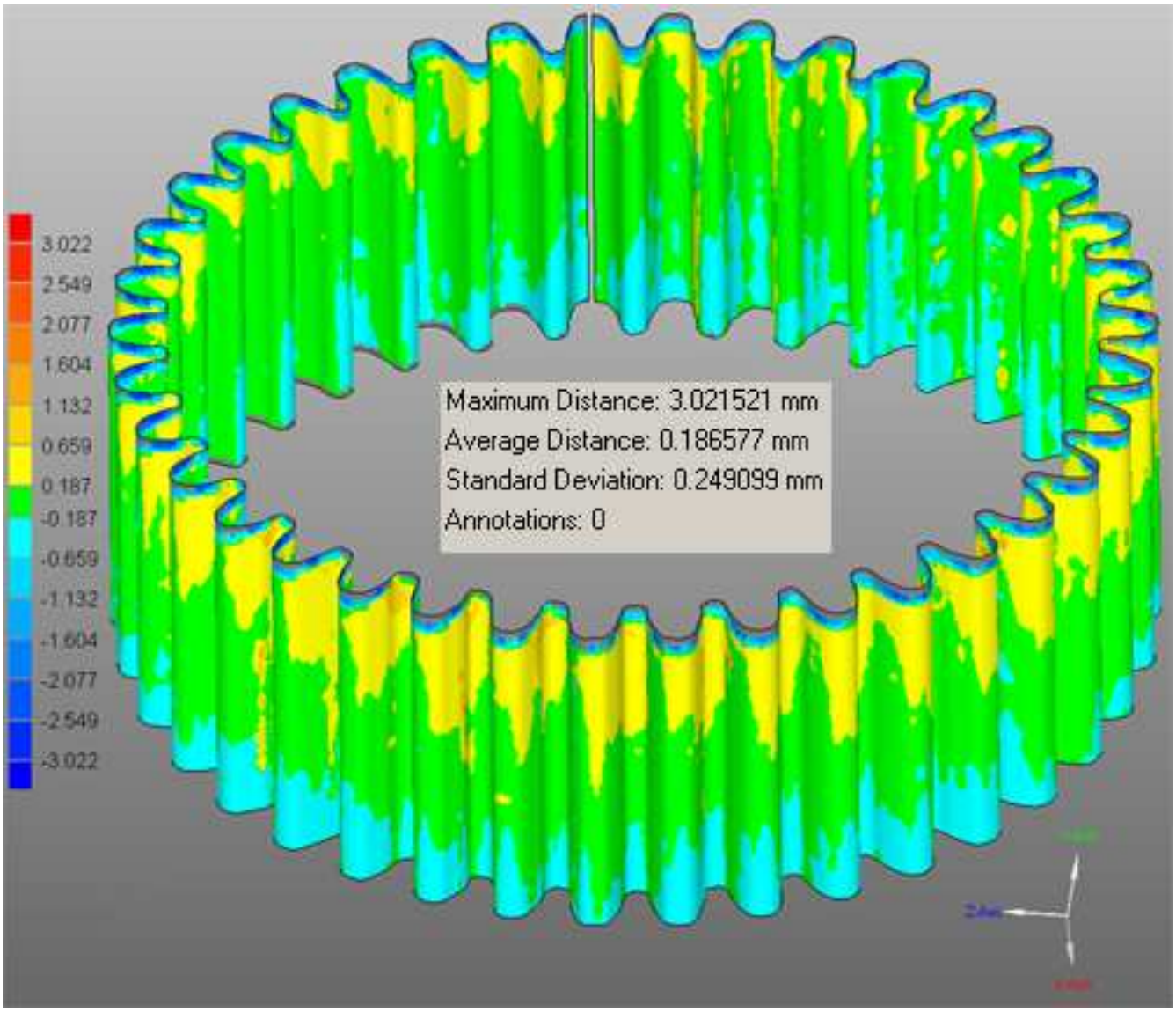

Figure 5.4: Gear outer surface deviation from measured points

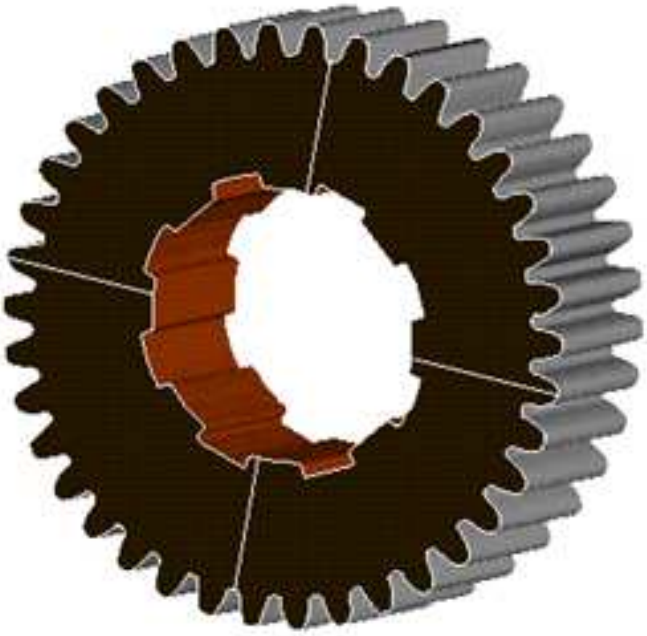

Figure 5.5: Reconstructed Gear

Points are projected to the plane and transformed to 2-D coordinates as shown in figure 5.2. The points are thinned and a subset is selected as shown in figure 5.3. Finally, the points are approximated using a least squares approximation to obtain a NURBS curve [36, p.410], and this curve is swept along a trajectory that passes through the object center, perpendicular and delimited by the two planes of the bounding box parallel to the profile curve. The swept surface for the inner part of the gear are obtained in a similar way, and the area between inner and outer is filled with a surface using the IDEAS CAD package.

Figure 5.4 shows the deviations between measured points and the swept surface and figure 5.5 shows the recovered gear.

5.2.2 Planar, Nonlinear Sweep Trajectory

The next example is a translational sweep where the trajectory is nonlinear. The scope of the approach is limited to sweeps where trajectories lie in a single plane, and where profile curves are perpendicular to the trajectory's plane. The original surface is generated using a skinning technique [36, p.457] to approximate a swept NURBS surface. The sweep is approximated by placing k profile curves along the trajectory and skinning across the profiles. Every kth profile curve is placed along the trajectory in a local orthonormal coordinate system where $\mathbf{T}(v)$ is the location of the origin, $\frac{\mathbf{T}'(v)}{|\mathbf{T}'(v)|}$ is the x axis, $y(v) = z(v) \times x(v)$ is the y axis, and the z axis is chosen as a unit vector such that $z(v)\dot{x}(v) = 0$ for all v. The test point cloud is created by randomly sampling points on this surface and perturbing the points by a maximum random error of 0.005 mm.

The profile curve is selected by first selecting three points on the edge of the object to create a slicing plane as shown in figure 5.6. The trajectory plane is selected by the operator who selects three points to approximate the trajectory plane (figure 5.7).

Finally, the profile and trajectory curves are projected to their slicing planes and approximated using least squares approximation to obtain NURBS curves, and the sweeping operation is reapplied giving the surface in figure 5.8.

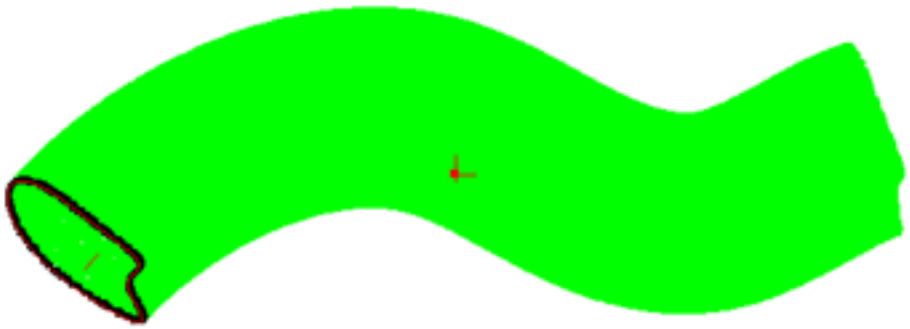

Figure 5.6: Plane containing profile curve created from three points selected by user.

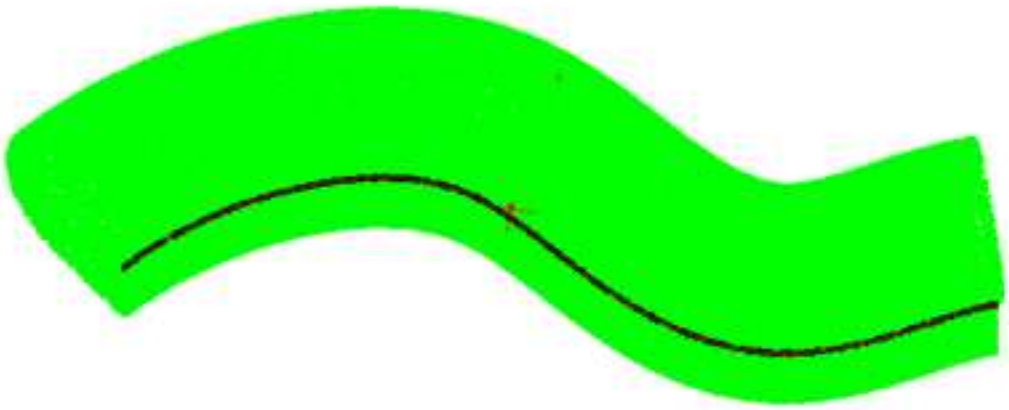

Figure 5.7: Trajectory curve

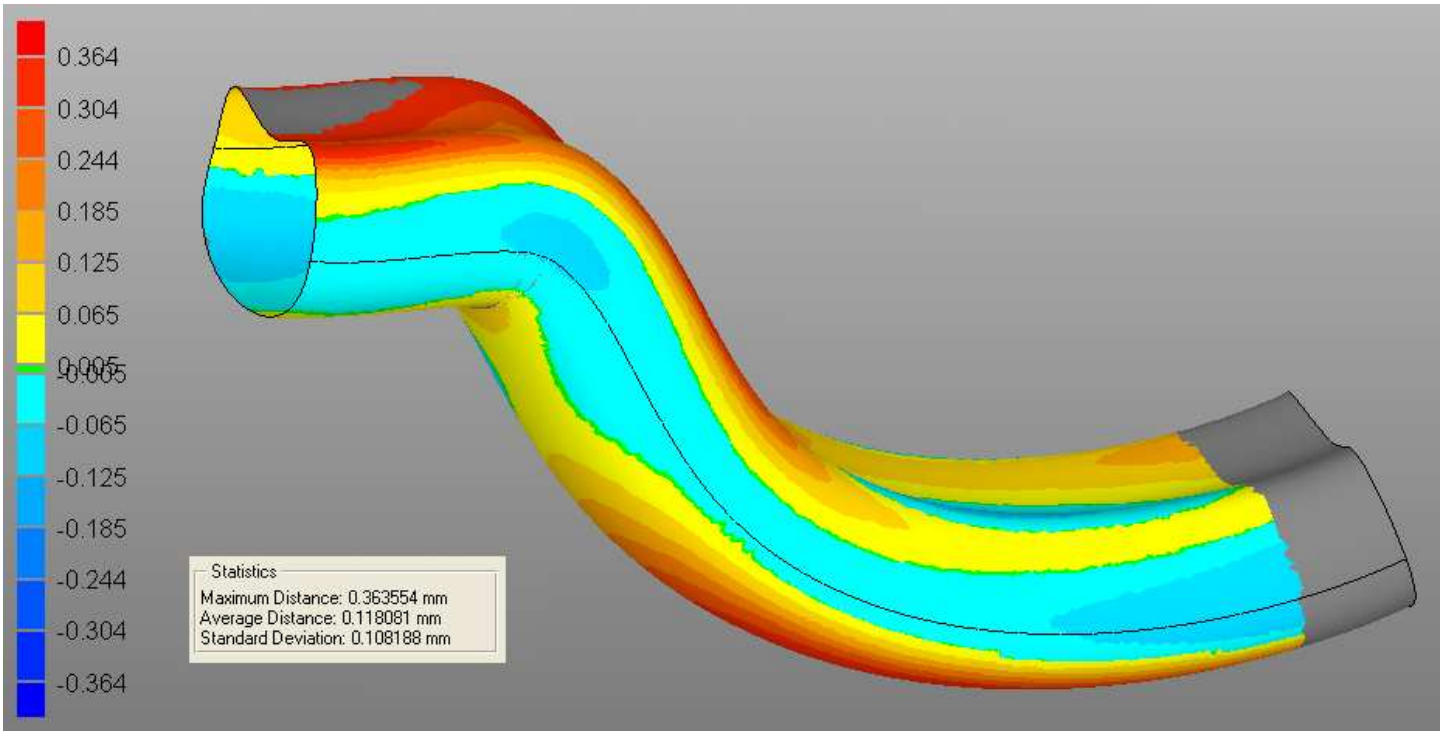

Figure 5.8: Recovery of sweep of profile along planar, non-linear trajectory

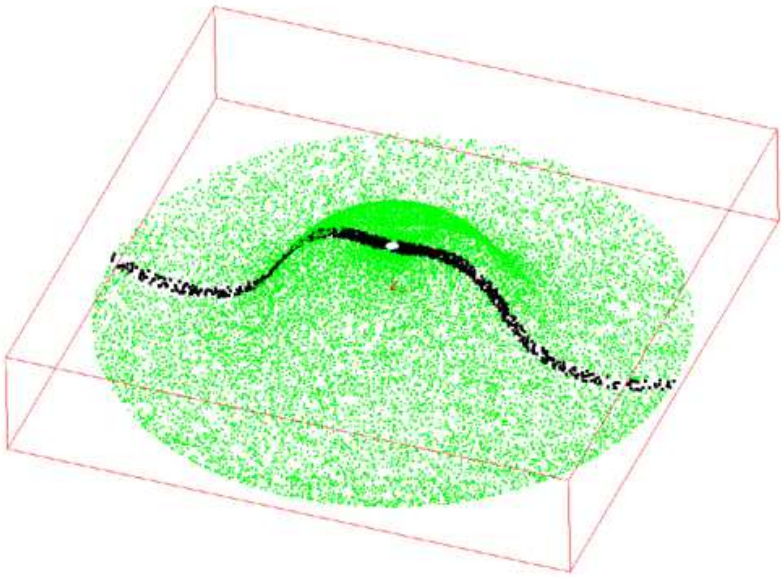

Figure 5.9: Profile curve for surface of revolution selected as plane containing vertical axis.

5.3 Rotational Sweep Recovery

The next example is a rotational sweep. The operator selects an axis of the bounding box to use as the axis of rotation. One of the planes of the bounding box containing the axis of rotation is used as the slicing plane for selecting a profile curve as shown in figure 5.9. The profile points are approximated with a least squares approximation and the resulting NURBS curve is rotated around the axis to produce the surface in figure 5.10.

5.4 Free-form Sweep Recovery

The next example is the recovery of a free form surface closed in one direction. The cloud of points is sliced into k section curves, each section being projected onto the slicing plane and approximated with a NURBS curve. The curves are made compatible as described in section 2.3, and a skinning technique [36, p.457] is then applied to the section curves to form the final surface. The number of slices required to accurately approximate the surface can be

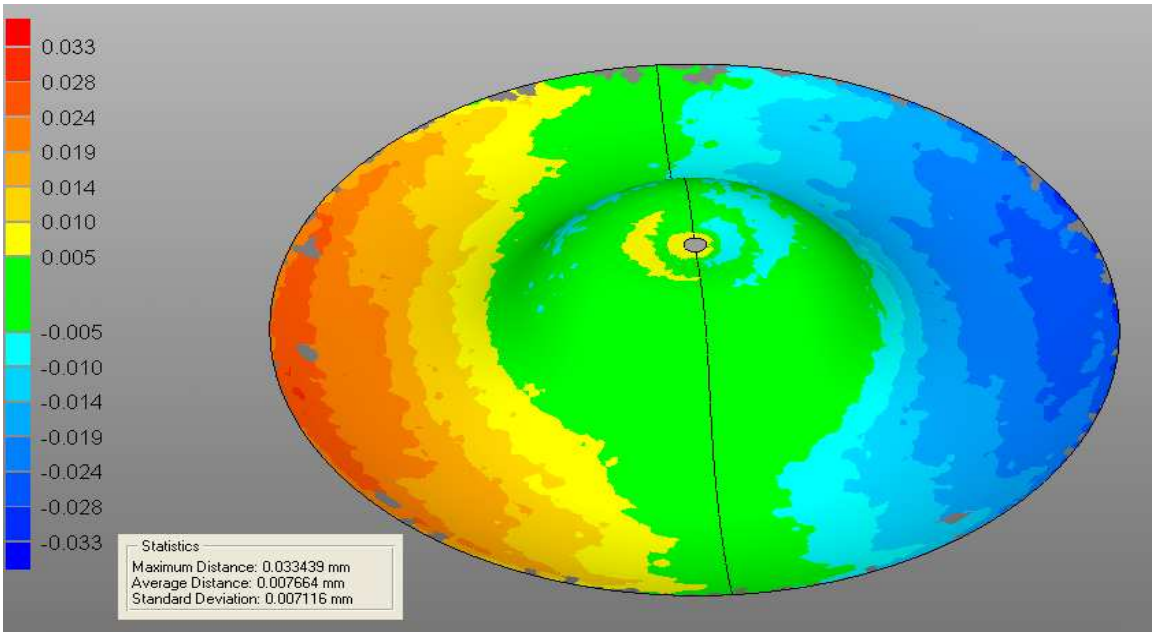

Figure 5.10: Surface of revolution generated from recovered profile curve.

determined by what the rapid prototype community calls adaptive slicing [31, 48, 43]. In adaptive slicing the thickness of each slice is increased until the maximum approximation error exceeds a chosen tolerance. Figure 5.11 shows the slicing planes perpendicular to the vertical axis of the point cloud's object coordinate system. Figure 5.12 shows the recovered surface.

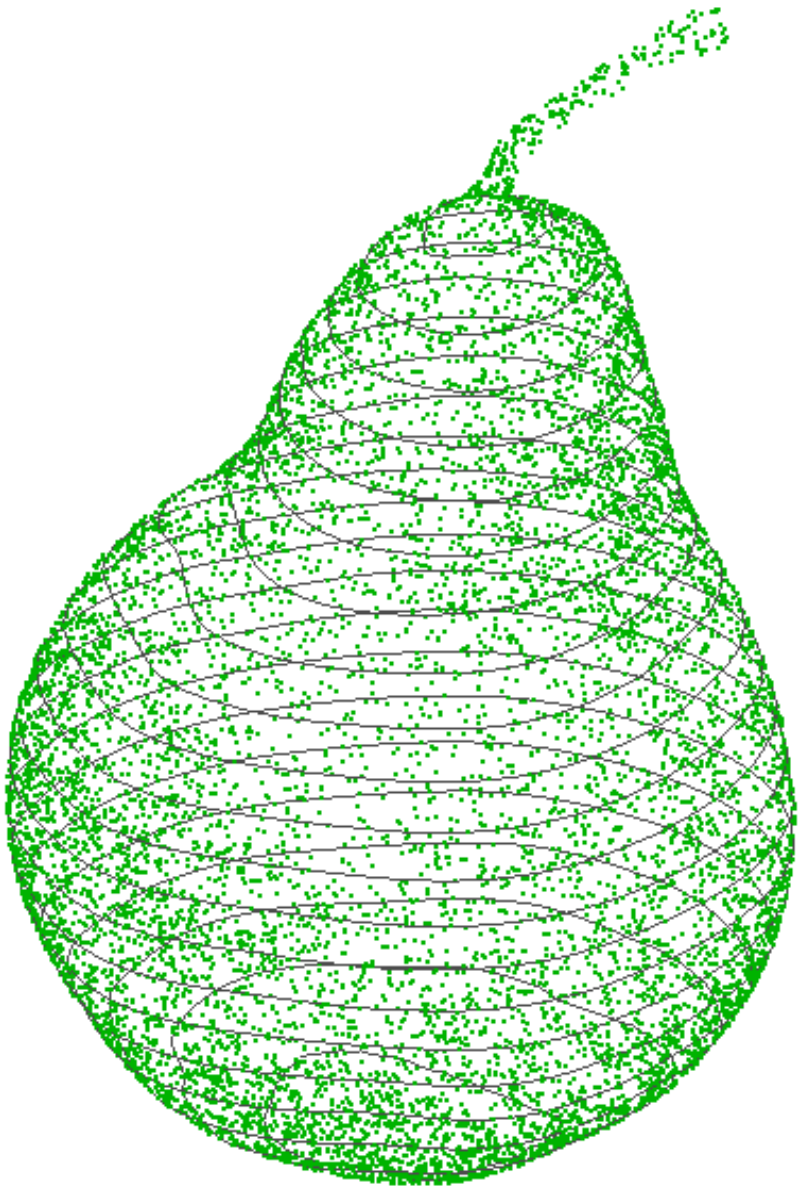

Figure 5.11: Scan points and section curves for free-form surface.

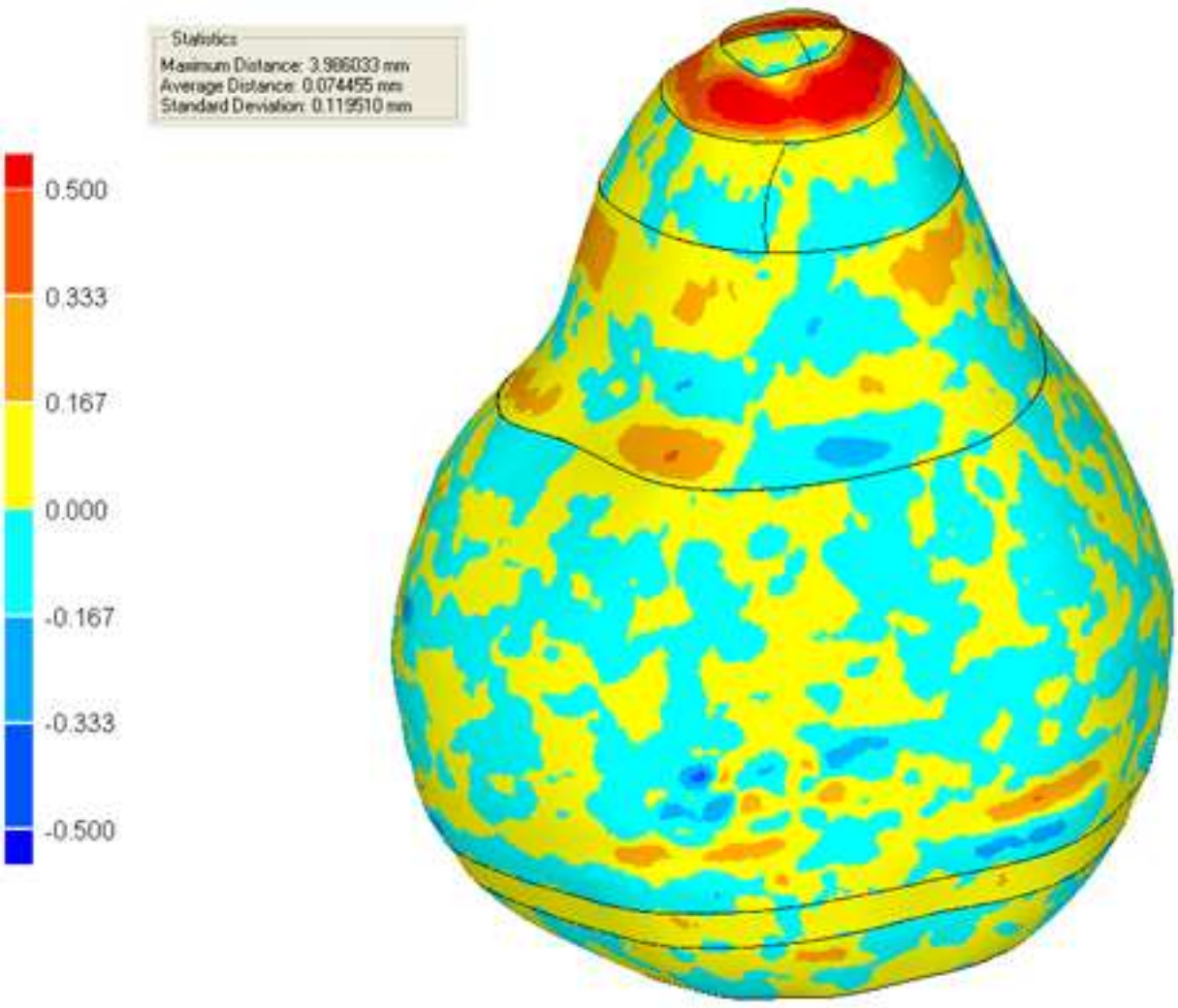

Figure 5.12: Deviation between free form surface (created by skinning section curves) and measured points.

Chapter 6

Summary and Future Work

This investigation of the reverse engineering process has presented an approach for recovering three types of swept surfaces. Translational sweeps are found by recovering a profile curve and a trajectory curve. Rotational sweeps are found by recovering a profile curve and an axis of rotation. Free-form swept surfaces are found by slicing the data into a set of contours along an axis and skinning a surface over the section curves. These recovery techniques all depend on the approach presented for recovering curves by projecting a set of points onto a slicing plane for curve approximation. This approach to recovering the three types of swept surfaces was demonstrated on a number of examples that included data taken from the scan of actual parts.

6.1 Surface Recovery Method

Translational Sweeps The approximation of the outer surface of a gear in figure 5.4 has an average deviation of 0.18 mm. The maximum deviation of 3.02 mm is due to the rounded edge on one face of the gear that could be further segmented and treated as a nonlinear translational sweep. The comparison of the surface with the measured points shows that areas at the top of the gear are slightly outside of the recovered surface, while areas along the bottom of the gear are slightly inside of the recovered surface. Using the feedback from the 3-D compare operation, the operator will have to judge make a judgment if the part is suitably represented as a translational sweep. The gear may not have been a perfect cylinder, or the error of the points may come from the position of the scanner high above the gear. The result

of the 3-D compare suggests that the profile is scaled along the trajectory. Future work should look at a method for finding this scaling factor along the trajectory.

In the case of the nonlinear sweep trajectory, the poor results are due to two major sources of error. First, it is very difficult for the operator to select the trajectory plane. Future work will automate the selection of the trajectory. Since the plane of the trajectory is perpendicular to the plane of the profile curve, the trajectory can be represented as a point and a normal where the point is the object center and the normal is a normal perpendicular to the profile curve's normal. This leaves one degree of freedom for the trajectory normal to rotate about the profile curve's normal. The proper orientation of the trajectory normal can be found by a binary search, rotating the normal about the profile normal until two planes parallel to the trajectory are tangent to the data. Second, the recovered surface is produced by placing k profile curves at local frames along the trajectory according to a trajectory tangent that may not accurately reflect the actual value of $\mathbf{M}(v)$ (see section 3.4). If these local profile curves are oriented to minimize the error between the profile curve and the point cloud, then the error of the sweep operation can be reduced. Future work would investigate setting up an optimization problem at every kth section curve along the trajectory.

Rotational Sweeps Figure 5.9 shows a regular cloud of points measured from a surface of revolution. The axis of revolution was estimated as the vertical axis of the object coordinate system determined through principal component analysis. Figure 5.10 shows that this was a fairly accurate approximation of the axis since the average deviation is 0.007 mm. Figure 5.10 also reveals the slight tilt of the axis as the points on the left are slightly above the surface and the points on the right are slightly below.

Point data may not always be so regular, especially data measured from a broken part. In these cases the object coordinate system will not reflect an accurate axis for the surface of revolution. Future work would set up an optimization problem to improve the approximated axis using a method similar to the work done by Lai and Ueng [19].

Free-form Sweeps Figure 5.11 shows a cloud of points measured from a free-form surface. The deviations shown in figure 5.12 suggest that the skinning of cross sections has room for improvement. Further work could

improve the parameterization of the cross sections through resampling and refitting [33].

6.2 Segmentation by Sweeping

Curve approximation by slicing is one of the operations listed as necessary for manual segmentation described in section 2.2. In future work the approach for recreating swept surfaces presented in this thesis can be applied within a reverse engineering system to implement a section by sweep operation. By selecting section curves and subsetting points in the point cloud near the swept surfaces, the point cloud can be narrowed down into a tree forming the part's boundary representation. Figure 6.1 shows a point cloud that is decomposed step by step into the sweep operations that make up the representation of the part. After each sweep, a portion of the points are removed and the remaining points become the input for the recovery of the next sweep. Finally, a tree of Boolean operations makes up the final part representation. Each leaf of the tree contains a set of points, a recovered surface, and an object coordinate system. Inner nodes are Boolean operations that either subtract or add the sweep to the final volume.

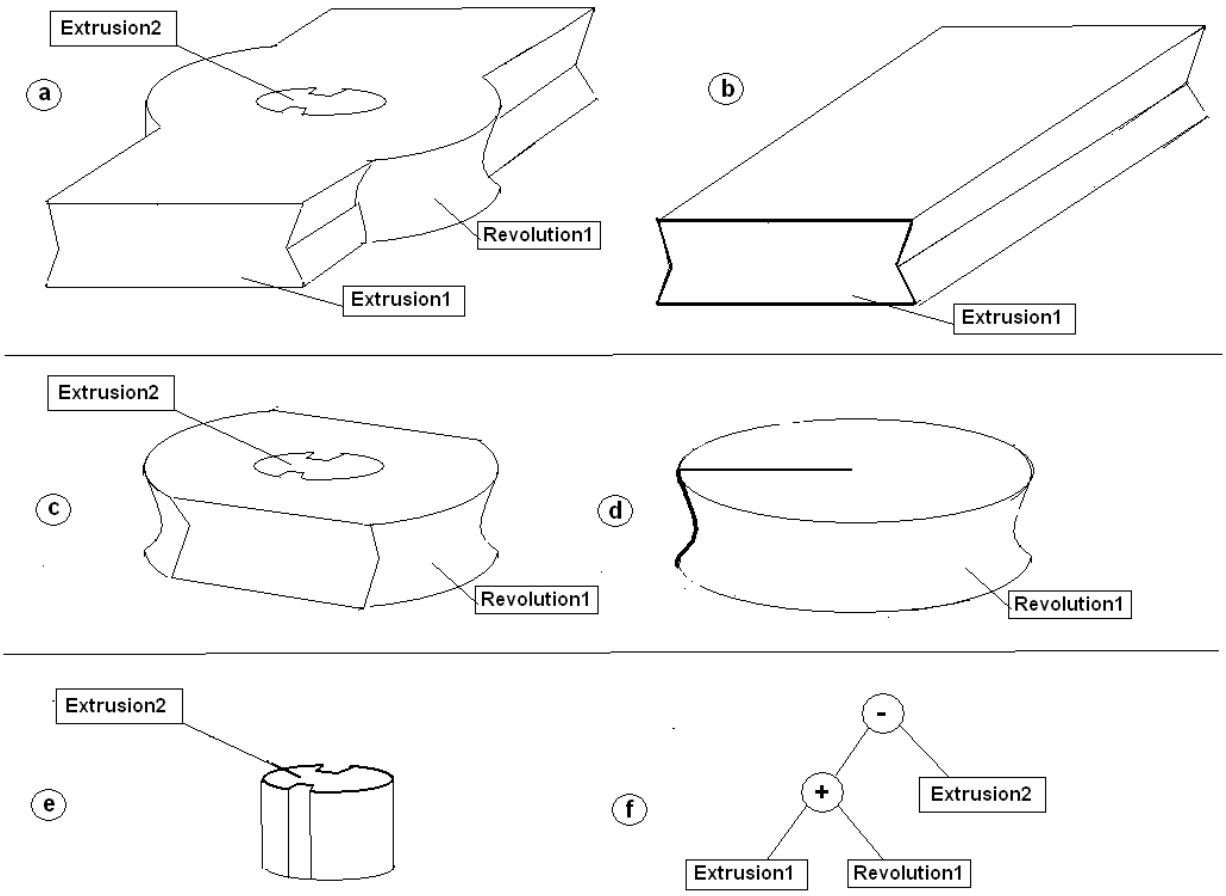

Figure 6.1: (a) Starting Point cloud. (b) Recovery of Extrusion1. (c) Points remaining after subtracting points near Extrusion1. (d) Recovery of Rotation1. (e) Points remaining after subtracting points near Rotation1. (f) Boolean tree describing boundary representation of segmented point cloud.

Bibliography

[1] Bae, S., Choi, B. *NURBS surface fitting using orthogonal coordinate transform for rapid product development*. Computer Aided Design, 2001;34:683-690.

[2] Benko, P., Marftin, R.R., Várady, T. *Algorithms for reverse engineering boundary representations models*. Computer-Aided Design. 2001;33:839-851.

[3] Besl, P. *Segmentation Through Variable-Order Surface Fitting*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1988;10(2):167-192.

[4] Besl, P., McKay, N.D. *A Method for Registration of 3-D Shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1992;14(2):239-256.

[5] Blackmore, D, Leu, MC, Wang, LP, Jiang, H, *Swept Volume: A Retrospective and Perspective View* Neural, Parallel & Scientific Computations, 1997;5:81-102.

[6] Blanc, C., Schlick, C. *Accurate Parameterization of Conics by NURBS*. IEEE Computer Graphics & Applications. 1996;16(6):64-71.

[7] Chen, Y., Medioni, G. *Object modeling by registration of multiple range images* Proceedings of IEEE conference on Robotics and Automations, Sacremento, April 1991:2724-2729.

[8] Chiyokura, H., Takamura, T., Konno, K., Harada, T. G^1 *Surface Interpolation over Irregular Meshes with Rational Curves*.

NURBS for Curve and Surface Design. Ed. Gerald Farin. Society for Industrial and Applied Mathematics, Philadelphia. 1991.

[9] Cignoni, P., Montani, C., Scopigno, R. *A fast divide and conquer delaunay triangulation algorithm*. Computer-Aided Design, 1998;30(5):333-341.

[10] Crossman, J. and Yoon, D.H. *Cutter swept surface construction using B-spline surface interpolation*. International Journal of Modeling and Simulation. 2001;21(4):292-300.

[11] Curless, B. *From Range Scans to 3D Models* Computer Graphics. Nov 1999;33(4)

[12] Fan, T., Medioni, G. *Segmented Descriptions of 3-D Surfaces*. IEEE Journal of Robotics and Automation, 1987;RA-3(6):527-538.

[13] Hoffmann, C. *Geometric and solid modeling: an introduction*. San Mateo: Morgan-Kauffman, 1989.

[14] Hoppe, H. et al, *Surface Reconstruction from Unorganized Points*. Computer Graphics, 1992;26:71-78.

[15] Hoppe, H. et al, *Piecewise Smooth Surface Reconstruction*. Computer Graphics, 1994;28:295-302.

[16] Edelsbrunner H. *Algorithms in combinational geometry*. New York: Springer-Verlag, 1987.

[17] Ganter, M., Uicker, J., *Dynamic collision detection using swept solids* ASME J. Mechanism, Transmission & Automation in Des. 1986;108:548-555.

[18] Krishnamurthy, V., Levoy, M. *Fitting Smooth Surfaces to Dense Polygon Meshes*. Computer Graphics, 1996;30:313-324.

[19] Lai, J., Ueng, W. *Reconstruction of surfaces of revolution from measured points*. Computers in Industry, 2000;41:147-161.

[20] In-Kwon Lee. *Curve reconstruction from unorganized points*. Computer Aided Geometric Design. 2000;17:161-177.

[21] Lengyle, Eric. *Mathematics for 3D Game Programming and Computer Graphics*. Charles River Media, MA. 2002.

[22] Leu, M.C., Park, S.H. & Wang, K.K., *Geometric representation of translational swept volume and its applications* ASME J. of Eng. Indust. 1986;108:113-119.

[23] Levin, D. *The approximation power of moving least-squares*. Mathematics of Computation. 1998;67:1517-1531.

[24] Liu, G.H., Zhang, Y.F., Loh, H.T. *Error-based segmentation of cloud data for direct rapid prototyping*. Computer-Aided Design, 2001;35:633-645.

[25] Ma, W., Kruth, J.P. *Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces*. Computer-Aided Design, 1995;27(9):663-675.

[26] Ma, W., Kruth, J.P. *NURBS Curve and Surface Fitting for Reverse Engineering*. Advanced Manufacturing Technology, 1998;14:918-927.

[27] McLain, D. Two dimensional interpolation from random data, The Computer Journal. 1976;19:178-181.

[28] David Meyers, Shelly Skinner, and Kenneth Sloan. *Surfaces from contours*. ACM Transactions on Graphics, July 1992;11(3):228-258.

[29] Milroy, M.J., Bradley, C., Vickers, G.W., Weir, D.J. G^1 *Continuity of B-spline Surface Patches in Reverse Engineering*. Computer-Aided Design, 1995;27(6):471-478.

[30] Milroy, M.J., Bradley, C., Vickers, G.W. *Segmentation of a wrap-around model using an active contour*. Computer-Aided Design, 1997;29(4):299-320.

[31] Pandey, P.M., Reddy N.V., Dhande, S.G. *Real time adaptive slicing for fused deposition modelling*. International Journal of Machine Tools and Manufacture, 2003;43:61-71.

[32] Park, H. and Kim, K. *Smooth surface approximation to serial cross-sections*. Computer-Aided Design, 1996;28(9):699-706.

[33] Park, H. and Kim, K. *A method for approximate NURBS curve compatibility based on multiple curve refitting*. Computer-Aided Design, 1999;32:237-252.

[34] Park, S.C. *Sculptured surface machining using triangular mesh slicing*. Computer-Aided Design, Available:internet, 20 May 2003, http://www.sciencedirect.com

[35] Piegl, L., and Tiller, W., *A menagerie of rational B-spline circles* IEEE Computer Graphics & Applications, 1987;9(5):48-56.

[36] Piegl, L. and Tiller, W. The NURBS Book, Springer-Verlag, Berlin. 1995.

[37] Piegl, L., and Tiller, W., *Algorithm for approximate NURBS skinning*. Computer-Aided Design, 1996;28(9):699-706.

[38] Piegl, L., and Tiller, W., *Symbolic operators for NURBS*. Computer-Aided Design, 1997:29(5):361-368.

[39] Piegl, L., and Tiller, W., *Parameterization for surface fitting in reverse engineering*. Computer-Aided Design, 2000;33:593-603.

[40] Piegl, L., and Tiller, W., *Algorithm for finding all k nearest neighbors* Computer Aided Design. 2000;34:167-172.

[41] Schneider, Philip J. and Eberly, David H. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, San Franscisco. 2003.

[42] Szeliski, R. and Tonnesen, D. *Surface modeling with oriented particle systems*. Computer Graphics, July 1992;26(2):185-194.

[43] Tyberg, J., Bohn, J.H. *FDM systems and local adaptive slicing.* Materials and Design, 1999;20:77-82.

[44] Várady, T. Martin, R. and Cox, J. *Reverse engineering of geometric models—an introduction* Computer-Aided Design. 1997;29(4):255-268.

[45] Wang, C. *Rapid Manufacturing* Available:internet. Aug 18, 2002. http://www.csa.com/hottopics/rapidman/overview.html

[46] Woodward, C.D. *Cross-sectional Design of B-Spline Surfaces.* Comput. & Graphics, 1987;11(2):193-201.

[47] Woodward, C.D. *Skinning techniques for interactive B-spline surface interpolation.* Computer-Aided Design, 1988;20(8):441-451.

[48] Wu, Y.F., Wong, Y.S, Loh, H.T., Zhang, Y.F. *Modelling cloud data using an adaptive slicing approach.* Computer-Aided Design, Available:internet, 20 May 2003, http://www.sciencedirect.com

[49] Yang, M., Lee, E. *Segmentation of measured point data using a parametric quadric surface approximation.* Computer-Aided Design, 1999;31:449-457.

[50] Raindrop Geomagic product literature.