FROM POINT CLOUDS TO TENSOR PRODUCT B-SPLINE SURFACES

by

Lavanya Sita Tekumalla

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

School of Computing

The University of Utah

May 2005

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Lavanya Sita Tekumalla

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Elaine Cohen
	Richard Riesenfeld
	Mike Kirby

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Lavanya Sita Tekumalla in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Elaine Cohen
Chair: Supervisory Committee

Approved for the Major Department

Christopher Johnson
Chair/Director

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

Being able to reverse engineer from point cloud data to obtain three-dimensional (3D) models is important in modeling. We present a new method to obtain a tensor product B-spline representation from point cloud data by fitting surfaces to appropriately segmented data. Point cloud data obtained by digitizing 3D data, typically presents many associated complications like noise and missing data. Our method addresses all these issues, works robustly in the presence of holes in the data and is straightforward to implement.

To My Parents

CONTENTS

ABSTRACT . . . . . **iv**

LIST OF FIGURES . . . . . **viii**

ACKNOWLEDGMENTS . . . . . **x**

CHAPTERS

1. MOTIVATION **1**

1.1 Problem Definition 2

2. PREVIOUS WORK **4**

2.1 Least Squares Fitting: Background 4
2.1.1 Weighted Least Squares 5
2.1.2 The Moving Least Squares 6
2.1.3 MLS Projection 6
2.1.4 Minimizing Perpendicular Distance 8
2.2 Scattered Data Approximation 9
2.3 Some Methods of Attempting to Deal with the Entire Model by Making a Network of Patches 9
2.4 Other Approaches for Scattered Data Approximation 10
2.5 Knot Placement 11
2.6 Hole Filling 13
2.6.1 Hole-filling Integrated into Reconstruction 13
2.6.2 Hole-filling as a Postprocess 13
2.6.3 Mesh-Based Methods 14
2.7 Smoothing 14
2.8 Parameterization 15
2.8.1 Parameterization for Curves 16
2.8.2 Parameterization for Surfaces 17
2.8.3 Parameterizing Triangular Meshes 18
2.8.4 Harmonic Maps 18
2.8.5 Conformal Maps 19
2.8.6 Parameterization for Fitting 20
2.8.7 Improving Parameterization: Handling Distortion in Scale 20
2.9 Commercial Solutions 21

3. FRAMEWORK **22**

4. SMOOTHING **24**
4.1 Boundary Smoothing 24
4.2 MLS Projection for Curves 24
4.3 Discussion 27
4.3.1 Handling Sharp Corners in the Boundaries 28

5. HOLE FILLING **32**
5.1 Steps in the Hole-Filling process 34
5.1.1 Make-Convex 34
5.1.2 Add-Vertices 36
5.2 Results 36

6. PARAMETERIZATION **40**
6.1 Domain Completion 45

7. FITTING **50**
7.1 Knot Placement 50
7.2 Fitting 51
7.2.1 Blending Local Fits 51
7.2.2 The Local Fit 52
7.3 Quantifying the Quality of Fit 57
7.4 Analysis of Blending Local Fits 59
7.5 Results 62

8. CONCLUSION **65**

9. FUTURE **66**
9.1 Reparameterization 66
9.2 Domain Specific Methods 66
9.3 Knot Selection 66
9.4 Hole Filling for Point Clouds 67
9.5 Handling Lack of Data 67
9.6 Continuity between Patches 67
9.7 Meshless Method 68

REFERENCES **69**

LIST OF FIGURES

2.1 MLS projection procedure 7
2.2 An example of a model that cannot be parameterized by projection . . . 17
3.1 The framework 23
4.1 An example of the smoothing process. 25
4.2 An example of the tangential noise present in the boundary that is not eliminated by normal smoothing. 25
4.3 An example of the tangential noise present in the boundary that is not eliminated by normal smoothing. 26
4.4 MLS projection procedure 27
4.5 An example of the curve smoothing process. 28
4.6 The 3D curve smoothing process illustrated on a nonplanar spiral 29
4.7 The wheel model after smoothing both the surface and the boundary. . 30
4.8 The curve smoothing process illustrated to smooth the boundary of the surface to remove tangential noise 30
4.9 A curve with sharp features that are not preserved when the entire curve is treated as a single piece. 31
5.1 The effect of holes in the fitting phase 33
5.2 The situations that can arise if the various checks are not performed . . 34
5.3 The various steps in the hole filling process. 35
5.4 The bottom of the teapot model smoothly filled 38
5.5 Hole filling carried out for a hole whose vicinity is nonplanar 38
5.6 Another example of the hole filling process 39
5.7 Hole filling for the wheel model 39
6.1 Mean-value coordinates 41
6.2 Wheel model parameterized using mean-value coordinates by fixing the boundary by chord length 42
6.3 Bean model parameterized using the harmonic maps with four corner points fixed. 43
6.4 Bean model parametrized by chord length 44
6.5 The domain completion process illustrated for the bean model 46

6.6 The domain completion process for the bean model 47
6.7 The influence of the domain completion process on the final surface . . . 48
6.8 The process of filling the domain and adding new points for incomplete surfaces. 49
7.1 Recursive domain decomposition 51
7.2 The domain decomposition process 52
7.3 Weighting function that windows the neighborhood. 54
7.4 Basis functions 54
7.5 A figure showing the control polygons of four local patches and the process of blending 55
7.6 A figure showing a single B-spline basis function and its influence over each patch 55
7.7 A figure showing the control polygons of two local patches and the process of blending 56
7.8 Results obtained for the curve case with various blending schemes 57
7.9 Control points when blending local fits 58
7.10 Comparison of the quality of fit of the global least squares fit method with the blending local fits method 60
7.11 The final fit obtained when using the global least squares fit method and the blending local fits method 60
7.12 Comparison of the quality of fit of the global least squares fit method with the blending local fits method by showing the error as a color map (red denotes regions of high error and green indicates regions of error) . 61
7.13 An example demonstrating the entire framework 63
7.14 The entire framework on the bean model 64

ACKNOWLEDGMENTS

I am extremely grateful to Prof. Elaine Cohen, my Thesis Supervisor, who first introduced me to this topic. I particularly thank her for her enormous patience during the early days, her continued interest in my work, and her constant guidance throughout the duration of my stay here. I am grateful to Prof. Rich Reisenfeld and Prof. Mike Kirby for their valuable comments and suggestions at various stages of my work.

I sincerely thank my parents for their constant support, encouragement and invaluable advice throughout my career. I especially thank them for their patience and commitment that helped me to make my dreams come true. I also thank my friends and roommates, with whom I held several enlightening dicussions, for their time and patience.

CHAPTER 1

MOTIVATION

CAD technology for mechanical design, engineering, and manufacturing is now playing an important role in production. The process of designing an aesthetic shape, which fulfills specified functional requirements is rather complex. Designing models by hand and sketches based on existing real world objects is a highly laborious process. Automating the design process in this area shortens the time of development and lowers the cost of development of a product, while at the same time increasing the reliability and the quality of the product. Also, converting real world objects into CAD models is extremely useful when a CAD model should be adapted to an already existing object by manipulating the existing shape.

The entertainment industry is another area where such modeling is widely used. Facial models and other organic forms, which are difficult to create otherwise, can be created by digitizing live humans or physical models. Modeling from scratch is slow and tedious in such situations. Now, automatic digitizing equipment like a laser scanner or a range scanner can be used to create high-resolution 3D models of live human models complete with color data. Therefore there is a widespread need to convert point clouds obtained from digitizing 3D physical data into computer models.

The common surface representations for 3D models are polygon meshes, NURBS and recently point sets. Each representation has its own merits and deficiencies. For instance, the polygon mesh representation has the advantage that it is extremely flexible and allows for local modification with fine detail. Point sets provide an efficient representation for huge data sets where the size of the model is far greater than the display device. NURBS surfaces have the inherent advantage that one can intuitively manipulate the overall geometry effectively and easily by changing the control points. Also they lend themselves to a more analytical treatment of the surface, meaning, we can readily evaluate the properties like tangents and curvature.

Hence NURBS are the industry standard in CAD/CAM applications and are the most adapted representation.

Thus, the problem of automating design for manufacture and production by converting real world objects into computer models is important. Digitizing 3D data and reverse engineering data to obtain 3D NURBS models that would result from *ab initio* design has numerous applications, and is a much explored area.

1.1 Problem Definition

3D data from real world objects obtained by scanning models can have several associated technical complications. These include

1. Noise: noise in the data can give us incorrect information about the geometry of the model during fitting. Therefore, it is necessary to eliminate noise before processing the data and fitting the surface.

2. Incomplete data: Data obtained from a laser scanner typically has missing pieces or incomplete data due to several factors such as occlusion, low reflectance. Such artifacts can lead to lack of data in some regions of the model causing numerical instabilities in the process of fitting and might lead to a wrong surface.

3. Holes: Holes in the surface may be caused by factors such as missing pieces in the original geometry and the process of segmentation. If fitting is performed without addressing the problem of holes, lack of data at the hole area might affect the area around the hole and we might lead to a wrong surface. Also presence of holes might cause rank deficiencies during the fitting process and it is difficult to parameterize the data with holes.

Apart from these problems associated with point cloud data, the task of reverse engineering point clouds to fit trimmed NURBS necessitates finding a suitable parameterization for the data, a knot placement strategy and a good surface fitting algorithm.

The aim of this thesis is to create a framework to reverse engineer point clouds and fit tensor product B-spline surfaces to facilitate making trimmed NURBS. Also

this thesis aims to reverse engineer and obtain NURBS surfaces that would result from an *ab initio* design, to facilitate the users to modify and manipulate the model.

In doing so this thesis hopes to deal with noisy data and data with holes and addresses issues such as finding a good parameterization, an automated knot placement strategy and an efficient fitting method.

Also, this thesis deals with a single patch of a segmented model that is homomorphic to a disc, with an arbitrary number of holes. Hence this work does not aim to deal with issues such as approximating very sharp features in the input and assumes that such features are not present in the input. Also this thesis assumes that an underlying mesh structure is available with the data to use better parameterization methods.

CHAPTER 2

PREVIOUS WORK

The problem of converting point clouds to trimmed NURBS has numerous applications. Though there has not been considerable work dealing with an integrated framework for such a process, several aspects of this problem have been thoroughly explored.

The reverse engineering problem can broadly be classified into semiautomatic and automatic methods. Also, it can be classified into methods in which there is one single NURBS patch or where there is an entire model by constructing a network of patches.

2.1 Least Squares Fitting: Background

The term fitting can normally mean either interpolation or approximation. Interpolation is the process of finding a function that passes through all the data and it requires that we have as many points as the degrees of freedom. Approximation, the process finding a closest possible function that follows the data, is done when there is more data than degrees of freedom.

The most traditional criterion for approximating data when we have more data than the degrees of freedom is the linear least squares method which minimizes the sum of the squares of distances of all points from the fitted surface. This method dates back to 1806 when Legendre suggested this method. In general, several other forms of error might be minimized other than the L^2 norm (For particular applications, it might be appropriate to minimize the L^1 or the L^∞ norm) .One of the reasons for the choice of minimizing the least squares error is that differentiating the square of the error yields a linear problem that can be readily manipulated.

Suppose there are $m+1$ points $P_i = (x_i, y_i, z_i)$, to fit a surface $s(x, y) = \sum_{i=0}^{n} a_i f_i(x, y)$ to this data, where $\{f_i\}$ are the basis functions and $\{a_i\}$ the corresponding coefficients.

$$\mathbf{A} = \begin{pmatrix} f_0(x_0,y_0) & f_1(x_0,y_0)\cdots & f_n(x_0,y_0) \\ f_0(x_1,y_1) & f_1(x_1,y_1)\cdots & f_n(x_1,y_1) \\ \vdots & \vdots & \vdots \\ f_0(x_m,y_m) & f_1(x_m,y_m)\cdots & f_n(x_m,y_m) \end{pmatrix}, X = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ an \end{pmatrix} \text{ and } B = \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_m \end{pmatrix}$$

The least squares approximation method proceeds by minimizing the L^2 Norm of the residual $(\sum_{i=0}^{m}(zi - s(x_i,y_i))^2)^{1/2}$ or $Min||AX - B||_2$

A solution can be found directly by solving the normal equations $A^TAX = A^TB$ by finding the pseudo-inverse of A, $(A^TA)^{-1}A^T$, or using the process of singular value decomposition of A .

2.1.1 Weighted Least Squares

The weighted least squares technique is a related method, which tries to minimize the least squares error and at the same time tries to give different levels of importance to points by associating a weight with each error term. The relative importance of the points in determining the fit can be based on the certainty of the data as suggested by [1] or on other criteria based on characteristics the problem. Sometimes, it might be required that certain points are approximated more carefully than others and hence these points are given more weight. Also, this method is common when dealing with noisy data, where the accuracy of each data point is known. Points with more certainty can be given more weight.

The weighted least squares fit attempts to minimize $\sum_{i=0}^{n}(z_i - s(x_i,y_i))^2 w(x_i,y_i)$.Let

$$W_k = \begin{pmatrix} w_k(x_0,y_0) & 0 & \cdots & 0 \\ 0 & w_k(x_1,y_1) & & \\ \vdots & & \ddots & \\ 0 & & & w_k(x_m,y_m) \end{pmatrix}$$

The normal equations can be written as $A^TW_KAX = A^TW_KB$ and solved in a way similar to the previous system. So,

$X = (A^TW_KA)^{-1}A^TW_KB$

The right weights must be determined during the fitting process, based on the domain knowledge and the context. A number of methods have been proposed [2, 3, 4] that have considered a nonlinear least squares problem with the knot positions as unknown parameters. These methods will be discussed in later sections.

2.1.2 The Moving Least Squares

One way of assigning weights in the weighted least squares method is to define the function with respect to a specific point (or from the point of view of a particular point), so that generally the points in the neighborhood of the point under consideration have more weight than the other points as described by Cleveland [5]. While weighting functions are chosen based on the problem at hand, the common choice for the weighting function that achieves this effect is the Gaussian function, $w_k(x_i, y_i) = e^{\frac{-[(x_i-x_k)^2+(y_i-y_k)^2]}{h}}$, where (x_k, y_k) is the *point of view*. Therefore a different fit is obtained every time the point of view is changed. The choice of h basically determines how many points around a particular point have a significant weight to contribute to the fit and is a parameter that has to be chosen based on the data. This method of weighing points is the basis for the MLS projection method.

2.1.3 MLS Projection

Alexa et al. [6] have advocated the use of points as rendering primitives due to the increasing complexity in size of polygonal models. Each polygon typically occupies less than a pixel on screen space and hence representing a surface as a set of points directly is an attractive alternative. Since a point set surface implicitly defines a surface, to facilitate their use, they provide methods for up sampling and down sampling point density for efficient rendering. Further, they give a projection procedure based on work by Levin [7] to project every point P near the surface onto the surface. The MLS surface is defined implicitly as the set of points that project onto themselves. The projection procedure is briefly described here.

For each data point, a neighborhood is obtained by taking the n closest points (KD tree data structure can be used to make the closest point search faster). First,

they find a local reference plane for the neighborhood and then find a local parameterization by projecting the neighborhood onto this reference plane.

If r is the point under consideration, let q be the projection of r on the local reference domain H (that is yet to be found). Then q can be written as $q = r + nt$, where t is the signed distance of the reference plane H from r and n its normal.

H is solved for by minimizing the perpendicular distance of each point in the neighborhood from H and weighing each error term by the distance of the point from q . In other words, they find n and t in such a way that $\sum_{i=1}^{N} \langle n, p_i - r - tn \rangle^2 \theta(||p_i - r - tn||)$ is minimized, where θ is a gaussian weighting function defined as $\theta(x) = e^{(-x^2/h)}$ and h controlls the standard deviation of the gaussian.

Since $q = r + tn$, the distance from q (instead of r) is used in the weighting function. This makes it a nonlinear optimization process, so Powell minimization is used to solve for H. The initial guess for the minimization process is obtained by setting t to 0 and solving the above equation to get an initial estimate of n.

Once the local reference plane H is obtained, a local coordinate system is created with q as the origin, and n as the z axis. Now a bi-quadratic polynomial g is fit to approximate the surface locally using the moving least squares technique around q by minimizing $\sum_{i=1}^{N} (g(x_i, y_i) - f_i)^2 \theta(||p_i - q||)$. This polynomial is evaluated at $(0, 0)$ to get the desired MLS projection.

At the thought of finding a local reference plane, the first option that comes to mind is to find a plane by doing a weighted PCA analysis to find the weighted least squares best fit plane that minimizes the perpendicular distance of each point to the

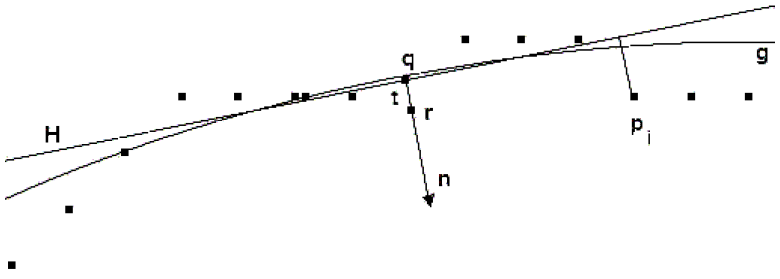


Figure 2.1. MLS projection procedure

plane weighed by its distance from the point r (see Figure 2.1). However such an approach does not result in a projection onto the surface. In other words, when the projection procedure is applied twice, the same point on the surface must be obtained in order to adhere to the definition of a point set surface. Hence the weights are based on the distance from q rather than r (see Figure 2.1).

2.1.4 Minimizing Perpendicular Distance

The desirable way of performing the least squares minimization is to minimize the square of the perpendicular distance of the data (shortest distance) to the fitted surface. This might be easy to formulate for simple cases, for instance a line and a plane. (Minimizing perpendicular distance of a point (x_1, y_1) to a line $y_i - mx_i - c = 0$, $Min \sum_{i=0}^{m} \frac{(y_i - mx_i - c)^2}{1+m^2}$, results in a quadratic expression (whose roots can be found easily). Similarly, the best fit plane passing through a set of points, say $Ax + By + Cz + d = 0$, can be obtained, by minimizing the square of the perpendicular distance $Min \sum_{i=0}^{m} \frac{(ax_i + by_i + cz_i + d)^2}{a^2+b^2+c^2}$ as the plane that passes through the centroid of all the points and whose normal as the eigenvector corresponding to the smallest eigenvalue of the covariance matrix.

To fit higher degree surfaces and parametric surfaces, the result obtained by minimizing the perpendicular distance from a plane cannot be extended and it results in more complicated expressions. Hence for higher degree surfaces, instead of minimizing the error calculated using the perpendicular distance $\sum_{i=0}^{m} d_i^2$ (where d_i is the perpendicular distance), we minimize $[z_i - f(x_i, y_i)]^2$. The advantage of doing the latter is that this later expression is simply quadratic in nature (unlike the perpendicular distance error) . Therefore the derivative is simply linear. Hence it is required to solve a simple linear system for the unknown parameters.

For parametric surfaces, the error that is commonly employed , is the sum of squares of the pseudo-distance which is defined as $||\gamma(u, v) - P_i||$. The pseudo-distance is greater than or equal to the perpendicular distance and hence, to obtain a more accurate approximation, the actual distance of every point to the surface must be minimized. Finding the actual distance from a point to a parametric surface is a nonlinear minimization problem in itself.

A method to minimize the actual distance by iterative reparameterization for B-spline curves has been discussed by Hoschek [8], Plass et al. [9] and O'Dell [10]. This will be discussed in more detail in later sections.

2.2 Scattered Data Approximation

With the advent of new and powerful scanners, surface reconstruction from scattered data has become an increasingly important problem and considerable work has been done in this area.

One way to deal with scattered data is to minimize a combination of the L^2 norm and the smoothing norm. The functional that is minimized has two terms. One measures the deviation η of the data points from the fit, and the other δ, measures the nonsmoothness of the fit d. It takes the form $\epsilon = \eta + p\delta$ where p is a parameter that controls the influence of each criterion. Dierckx [2] generalized this idea from the univariate case and derived a suitable smoothing norm for bivariate splines.

Hormann et al. [11] present a method to approximate scattered data that is triangulated using a single tensor product B-spline patch or using hierarchical B-splines. They parameterize data by imagining the entire mesh to be a system of springs and obtaining a configuration that minimizes the potential energy of the system. They minimize a functional that is a combination of the least squares distance and a fairing term based on data dependent thin plate energy [12] in the fitting step.

Hoschek et al. [13] give a framework for reverse engineering point clouds to get trimmed NURBS in which they too use a fairness term while minimizing error for fitting. However, they parameterize data by projection and they do not take any special measures to deal with holes.

2.3 Some Methods of Attempting to Deal with the Entire Model by Making a Network of Patches

In 1996 Eck et al. [14] described a method to construct B-spline models of arbitrary topological types, by constructing a network of B-spline patches and a method of joining them with G^1 continuity. They use harmonic maps for parameterization. In this approach, a large number of B-spline patches are needed to represent

a model. Krishnamurthy et al. [15] presented another procedure, in which the user interactively segments the data into several patches by painting the patch boundaries. The method then parameterizes the data by resampling each irregular polygonal patch into a regular grid of points, and does an unconstrained gridded data fit of a B-spline surface to each gridded patch. In 2000 Park et al. [16] described a method in which they initially approximate the model using K-means clustering and then construct a NURBS patch network using a hierarchical graph representation.

Gregorski et al. [17] decompose the given point set into a quad tree-like data structure known as a strip tree. They use the strip tree to fit a set of least squares quadratic surfaces to the data points at the lowest level in the tree. These quadratic surfaces are then degree-elevated to bi-cubic surfaces and blended together to form a set of B-spline surfaces that approximates the given point set. They, however, uniformly divide their domain and hence introduce many unnecessary knots. Their method fails for complex geometry as they parameterize each individual patch by projection.

Though these methods deal with a more difficult problem of building the entire model, unlike this thesis, which deals with a single patch, none of these methods deals with holes and missing data.

2.4 Other Approaches for Scattered Data Approximation

Many people have considered the problem of constructing an implicit surface from scattered data. Hoppe et al. [18] obtain a signed distance function by examining the neighborhood of each point and finding the local tangent plane and normal orientation at each point. Several people have explored approximating scattered data using the radial basis functions(RBF) approach [19, 20]. The advantage of the RBF approach is that a compact representation of the surface can be obtained by solving a system that is easily invertible. Further, RBF approximations conveniently deal with holes and noise in the data. The implicit surface obtained can be easily polygonized for rendering.

Hierarchical B-splines [21] have been a popular choice to represent surfaces due to their ability to allow for local refinement and several people have investigated scattered data approximation using such a representation [22, 11]. Pfeifle et al. [23, 23] fit triangular B-splines to scattered data. This method has some advantages over tensor product B-splines like lower degree for the same smoothness and more flexibility to model arbitrary topologies due to the lack of the requirement for a rectangular domain.

However, tensor product B-splines are widely used and are a standard representation in the CAD industry. This thesis focuses on fitting tensor product B-splines to point clouds.

2.5 Knot Placement

When dealing with B-spline surfaces, there is the additional task of determining the knot vector. A knot vector is a monotone increasing sequence of real numbers that determines the intervals over which the B-spline basis functions are defined.

Knot placement has a significant effect on the resulting surface. Hence care must be taken while selecting the knot vectors. Having too few knots results in a bad approximation and having too many knots needs more computation power and can result in unwanted oscillations. Also a rigid scheme with uniform distribution of a fixed number of knots often results in a poor fit. Hence, the adequate number of knots must be adaptively placed at the right locations in order to capture the shape appropriately. Several attempts [4, 3, 2] have been made an attempt to consider the least squares problem as a nonlinear optimization problem where the position of the knots is also optimized along with the control points. These methods proceed by first separating the linear and the nonlinear aspect of the problem (for a given knot sequence, the problem is linear in nature). Then they consider the nonlinear optimization problem of finding the right knot sequence. deBoor and Rice [4] first tried a nonlinear approximation to fit cubic splines by starting with an initial set of knots and varying the knot locations one at a time in a carefully chosen interval.

Jupp [3] made a profound analysis of the free knot problem which points out that the methods to solve the free knot problem are hindered by the fact that the search for

the global optimum leads to many stationary points and that minimization algorithms showed poor behavior at the boundary leading to situations with almost coinciding knots. In order to mollify these problems, he proposes a transformation of knots that increases the likelihood of convergence to a global minimum. However, these methods deal with placing the knots at the best locations, but do not address the problem of choosing an appropriate number of knots. Also the initial knot sequence plays an important role, so it is difficult to determine a good knot vector to begin with.

Dierekx [2] approximates data starting with a knot vector with no internal knots and iteratively inserts knots. The method introduces a new knot each time at the middle of the domain and use a nonlinear process to find its optimal position. The knot insertion process is stopped based on the root mean square residual. A problem with this method is that each knot insertion operation is a nonlinear optimization problem. Also, the general drawback of this class of methods is that the nonlinear optimization process is computationally expensive. Baussard et al. [24] describe an iterative procedure to insert and delete knots adaptively. They attempt to to deal with the 2D inverse scattering problem and their approach is based on approximating the unknown object with B-spline basis functions defined over a small knot vector to reduce the size of their problem (which involves a nonlinear inversion procedure). They perform knot insertion, based on curvature, by computing a normalized mean curvature map of the entire object and thresholding the object to determine areas of high curvature. To overcome the problem encountered when there is a continuous high-curvature region, they introduce a minimum spacing between pairs of inserted knots. To delete knots, they assign a weight to each knot based on the difference in the least squares error with and without the knot and remove all knots that are redundant (whose weights are below a certain set minimum). This process is repeated until no more knots can be extracted. They iteratively alternate between the insertion and deletion steps for a fixed number of times, or until the change in the object due to the new knot vector is below a minimum. Although this method is well suited for the problem on which they focused, it is inefficient when used for the purpose of creating models in CAD applications due to its large time complexity.

2.6 Hole Filling

While some work has been done to perform hole filling for meshes, most of the previous work on hole-filling is based on constructing an implicit surface from the data that interpolates the hole.

2.6.1 Hole-filling Integrated into Reconstruction

A significant part of such research, integrates the process of hole filling into surface reconstruction. In [25], a surface reconstruction technique is presented based on level sets, that uses a variational approach and evolves the surface until it approximate the data adequately. This method inherently addresses the problem of holes. Also, most of the methods that are based on finding an implicit representation are based on finding a signed distance function.

In [26], a volumetric method is presented to obtain cumulative weighted signed distance function from multiple aligned range images, in which the authors address the problem of hole-filling by space-carving to obtain watertight models. This hole-filling method, however, suffers from producing unwanted geometry in certain cases and aliasing artifacts that they eliminate by postfiltering the reconstructed mesh by weighted averaging. In [27] poly-harmonic radial basis functions are used to fill holes by fitting an RBF to the signed distance function and extracting the iso-surface.

2.6.2 Hole-filling as a Postprocess

Treating hole filling as a postprocessing step has the advantages that the hole filling process can be used independently with any reconstruction method and can handle holes that arise from situations other than scanning defects. Also such a method can focus on operating near the vicinity of the hole and reduce computation time. Some work has been done on extending image inpainting [28] to surfaces that treats hole-filling as a postprocessing step. In [29] a method using volumetric diffusion is presented that obtains a volumetric representation of the data, a 3D grid of values of signed distance function, and fills holes using a diffusion process of alternate blurring and compositing operating only in the vicinity of holes. In [30] another slightly

slower method is presented, based on image inpainting, that uses a system of coupled anisotropic partial differential equations.

In [6], the MLS projection is used for adequate resampling of points in a point cloud for rendering purpose. However, the author focuses on efficiently dealing with undersampling and oversampling in the input and not on hole-filling.

2.6.3 Mesh-Based Methods

In [31], a mesh-based hole-filling algorithm is presented. In this method, a best fit plane in the vicinity of the hole is found, and data are parameterized by projecting the points orthographically onto this plane. Then data are sampled in the parametric domain and the moving least squares technique is used to interpolate data at these points. This approach is efficient to implement. However it can handle only holes of simple geometry that resemble a plane since it relies on parameterizing the vicinity of the hole by orthographic projection onto a plane. Although our method is similar to this method, we use a local technique that can handle holes with nonplanar geometry.

2.7 Smoothing

Considerable work has been done on denoising point clouds and triangular meshes. Most of these methods are local in nature and are based on image processing techniques. Smoothing point clouds is a simpler problem than smoothing triangular meshes due to the absence of connectivity constraints. While dealing with triangular meshes, care must be taken to preserve the connectivity and topology. In [6, 32, 33], an attempt is made to smooth point clouds, while [34, 35, 36, 37, 38, 39, 40, 41, 42] work with triangular meshes.

Also, denoising algorithms can be classified into two types as those that attempt to preserve sharp features and those that do not. In [34], Taubin attempts to reduce mesh smoothing to low pass filtering and gives an isotropic method based on Fourier analysis to smooth polyheadral surfaces of arbitrary topology. This algorithm is linear and efficient unlike earlier methods that are based on the optimization of a fairness norm and take considerable computation time.

Some research has been done[35, 36, 37, 38] on feature preserving methods and iterative methods based on an anisotropic diffusion. Work has been done based on filtering the normals of a mesh and modifying the mesh vertices to match the normals. See [39] for a detailed discussion on this topic. In [40], a locally adaptive Weiner filtering approach is used to smooth meshes and in [41], a wavelet based approach is proposed. In [42], a noniterative feature preserving mesh smoothing algorithm is presented based on robust statistical estimations, that treats sharp features as outliers relative to one another to preserve features.[43] present another simple and efficient bilateral mesh denoising algorithm using local neighborhoods.

In [6], the MLS projection is used to obtain a smooth manifold surfaces from noisy point clouds and [32] use a similar method and attempt to preserve sharp features using robust statistics based estimates. In [33], the authors propose a fairing method for point sets, based on anisotropic geometric mean curvature flow.

2.8 Parameterization

A parameterization can be defined as a mapping for every point of a surface from a suitable domain. The most common domain used is a plane, though other domains like a cylinder, sphere may also be used. In other words, a planar parameterization for a surface S is a mapping $f : D \rightarrow S$ and where $D \subset \Re^2$, $S \subset \Re^3$ and it maps every point T(u,v) on the parametric domain to a point P(x,y,z) on the surface. The parameterization problem is important in our case as it affects the quality of the final fit we get. The required condition for a parameterization is that the mapping must be a bijection (one-to-one and onto).

A mapping is said to be isometric or length preserving if the length of an arc in S is the same as the length of the arc in its domain D. While such parameterizations are ideal, they do not always exist. Surfaces that can be isometrically mapped to a plane fall into a category called 'developable surfaces'. A developable surface is a surface with 0 gaussian curvature everywhere. Equivalently, through any point on the surface, there passes a straight line, wholly contained in the surface. Cylinders and cones are common examples of such surfaces. A desired property of a parameterization is that it is as close to an isometry as possible. Let J be the Jacobian of the surface, when

parameterized using a mapping function f. Then J^TJ is called the metric tensor M. For a planar mapping, an isometric parameterization results in a metric tensor $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. When the metric tensor has the form $\begin{pmatrix} K & 0 \\ 0 & K \end{pmatrix}$for some K, the map is said to be conformal or angle preserving. Finally when the metric tensor has a determinant 1, the map is said to be area preserving or equiareal. Another way to express isometry is to say that if for each (u, v), the tangent vectors to the iso-u and iso-v curves passing through S(u, v) have a norm that is unity, the mapping is isometric. The mapping from the domain of the surface U,V to the actual surface is said to be conformal on the other hand, if the tangent vectors to the iso-u and iso-v curves passing through u,v are orthogonal and have the same norm. Conformal maps are angle preserving maps and they have the property that they are locally isotropic. For instance, a circle in the parametric domain maps to a circle on the model.

Another class of maps are harmonic maps that satisfy the Laplace equations $\delta u = 0$ and $\delta v = 0$ (where δ is the Laplace operator $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$).

2.8.1 Parameterization for Curves

The parameterization problem is easier for curves than for surfaces. The arc length parameterization or the unit speed parameterization gives an isometric mapping from a curve to a straight line. A common approximation that is made to achieve arc length parameterization is the chord length parameterization [44] (when we have an ordered set of points that form the curve). Here the euclidean distance between adjacent points is taken as the distance between the parameter values of adjacent points. This is not an isometric mapping, since the arc length is different from the chord length. However, it is better than a uniform parameterization for most cases.

O'Dell et al. [10] point out that chord length parameterization is not suitable when the curve varies greatly in curvature and they give a variable speed parameterization technique that is data dependent and is based on estimated tangent length and curvature. However, this consideration is not very important for the kind of problem that is the focus of this thesis. This thesis focuses on finding a NURBS representation to a segment of the entire data, which implies that there is no possibility of coming across sharp edges.

Figure 2.2. An example of a model that cannot be parameterized by projection

2.8.2 Parameterization for Surfaces

The problem of extending the parameterization techniques from curves to surfaces is nontrivial. With the assumption that the density of points at hand is high, the parameterization problem is easy to solve in a small local patch of the input. However, and a global parameterization for the entire object has no ready solution and an optimal parameterization still remains an unsolved problem.

The simplest and the traditional way of parameterization has been to choose a plane and project all the points onto the plane to use the projections as parameter values as in Ma et al. [45]. To employ projection, it is necessary to try to find a plane over which the projections of the points are single-valued. The best fit PCA plane is probably a good guess that can be considered, though it is not guaranteed that it results in a single valued map for all kinds of geometry.

The general problem with projection is that the mapping may not be single valued and more than two points might have the same projection when we have complicated geometry to parameterize. Even in a simple object as the following diagram shows, projection approach might not work. At any point of time more than half the sphere cannot be parameterized using projection. Also projection is neither angle preserving nor area preserving

Several methods exist that fix the boundary of the parameterization and proceed by saying that every interior point must be parameterized such that it is a convex combination of its neighbors in the parametric domain (which basically says that each

interior point must lie in the convex hull of its neighbors). These methods are broadly termed as 'convex combination maps'.

Floater [46] first described such a method to find a piecewise linear map from a point cloud to a plane by mapping the boundary of the point cloud to a convex polygon on the plane. His method proceeds as follows. For each interior point P_i, the method chooses a neighborhood of points that are close to it, say, and chooses a set of strictly positive weights such that $\sum_{j \subset Ni} \lambda_{i,j} = 1$ and says $Ui = \sum_{j \subset Ni} \lambda_{i,j} U_j$. This leads to a linear system which is solvable under the mild constraint that the neighborhoods are chosen in such a way that all points are boundary connected. He further suggests ways in which these weights (λ_i's) can be chosen. They can be chosen to be equal for all points in the neighborhood ($1/n_i$ where n_i is the number of points in the neighborhood) or they can be chosen to be proportional to the inverse of the distance from the point

This method ensures that the mapping obtained is a bijection. However, it does not try to minimize any quantifiable form of distortion discussed above.

2.8.3 Parameterizing Triangular Meshes

With triangular meshes, more information is available than a mere point cloud. The connectivity information is available and hence more information can be obtained about the neighborhood of a point. So more intuitive methods for parameterization can be formulated. Considerable work has been done in this area as polygonal meshes are a popular way of representing surfaces. It is useful to examine some of these methods.

Floater [47], with the aim of extending chord length parameterization to the bivariate case, proposed shape preserving weights for convex combination maps. Hormann et al. [11] proposed a method to parameterize the data by visualizing the mesh as a system of springs.

2.8.4 Harmonic Maps

These methods provide suitable weights for convex combination maps so that the resulting parameterization approximates a harmonic map by a piecewise linear map.

In 1995 Eck et al. [28] proposed a way to obtain a harmonic mapping for parameterizing meshes. If the surface is imagined to be a triangular mesh of springs and an attempt is made to flatten it onto a plane, the result of their method tries to minimize the potential energy of the spring system, which is given by the energy functional $E_{harm}[h] = 1/2 \sum_{(i,j)\subset Edges} w_{i,j}||h_i - h_j||^2$ where h_i and h_j are the parameter values of P_i and P_j. However, the weights obtained thus might not always be positive. To deal with this problem, when a negative weight is obtained, the weights are set to a global constant.

Recently, Floater [48] described a method to obtain harmonic maps using his mean value coordinates, which was motivated by the fact that harmonic maps satisfy the mean value theorm. This method does not suffer from the problem of negative weights.

2.8.5 Conformal Maps

In 1999 Hormann et al. [49] presented a nonlinear technique 'MIPS' (Most Isometric Parameterizations) that tries to maximize the conformality of piecewise linear mappings of a triangle on the model to a triangle in the parametric domain and positions each vertex after a minimization of a rational quadratic function.

Levoy et al. [50] describe a linear method to obtain conformal maps for triangular meshes. They show how the conformality problem can be turned into an unconstrained quadratic minimization problem. Further, unlike other methods where the boundary of the mesh must be fixed (since the u are v parameters of the parameterization are obtained by solving two independent systems, and are indirectly coupled), Levoy et al. construct a system where the u and v parameters are connected by a single global system and hence do not require that the entire boundary be fixed. In fact they obtain a parameterization by fixing just two vertices of the boundary.

Sheffer et al. [51] presented another nonlinear method that formulated the parameterization problem in terms of angles alone. Their method attempts to obtain a conformal map and does not require the boundary to be fixed. As a result of the optimization problem, they obtain a set of angles in the parametric domain that are used to compute the actual vertex positions.

2.8.6 Parameterization for Fitting

All the above approaches have been formulated with applications such as texture mapping and morphing in mind. When the problem of fitting B-spline surfaces for CAD models is considered, we need the parameterization to be intuitive in some sense. Often times, the end user of a CAD system does not have any knowledge about the underlying representation of the object. Hence some additional constraints must be enforced, so that the parameterization actually depicts the geometry of the object. When the control mesh is easy to understand, editing the object can be done in an easy and intuitive manner. For instance, if there are any sharp corners in the boundary of the object, it is intuitive for these sharp corners to be depicted in the parameterization as well. While it is desirable to have a general framework for parameterizing any object, sometimes, domain specific methods might better serve the purpose.

Therefore, for our purpose, it is desirable to fix the boundary of the object to a convex polygon and at the same time preserve the general shape of the boundary. As a consequence of this requirement, the boundary of the parameterization must be made as close to the actual shape of the boundary as possible. Though some of the above methods described above automatically fix the boundary, they either do not take the shape of the actual boundary into account or are nonlinear in nature and are computationally expensive.

Also, tensor product B-splines need a rectangular boundary. Hence there is a need to obtain a rectangular domain after the boundary is fixed.

Floater [52] attempted fitting tensor product B-spline surfaces by mapping the boundary of the object directly to a square in the parametric domain. However, such an approach is not suitable for the problem under consideration for the above mentioned reasons.

2.8.7 Improving Parameterization: Handling Distortion in Scale

Sander et al. [53] describe the geometry stretch metric, which attempts to prevent the under sampling of geometry when a model is mapped to a parametric domain. They consider the Jaccobian of the mapping function from a triangle in the model

to that in the parametric domain and the largest singular value and the smallest singular value of the Jacobian represent the largest and smallest length obtained when mapping unit length vectors from the parametric domain to the surface.

They define the stretch metric for a triangle as $\sqrt{\frac{1}{2}(\Gamma^2 + \gamma^2)}$. They also outline a method to move each vertex in the parameter space to a point that minimizes stretch. Though this method gives very promising results, this involves a nonlinear optimization process to find the optimal location of each vertex.

2.9 Commercial Solutions

Many commercial softwares are also now available that provide for the user to interactively create surfaces from the digitized data. Some of the popular ones are, Imageware Surfacer (Metrix Software Solutions Ltd.), MedCAD (Materialise USA), Alias Studio (Alias@) and Paraform, Scan (Metris) These software solutions generally create a patch-wise approximation of polygonal data. Geomagic Studio (Raindrop Geomagic) is another software with reverse engineering capabilities that produces polygon and NURBS models of the surface, handling small holes in the data. While this software is geared at producing watertight surfaces for manufacture and production, the focus of this thesis is reverse engineering to obtain surfaces that would result from an *ab initio* design.

CHAPTER 3

FRAMEWORK

The aim of this thesis is to generate trimmed NURBS surfaces from point clouds. The focus of this thesis is on fitting an entire object of moderate complexity or a single patch of a complex object. In order to deal with all the issues discussed, this takes a multistage approach, as shown in Figure 3.1. The work can be briefly classified into the following steps.

1. Smoothing: First, a preprocessing step of smoothing the data is used to deal with noise.

2. Filling Holes: The MLS approximation is used to fill holes in order to find a global parameterization and prevent problems due to lack of data near the hole in the fitting phase.

3. Parameterization to obtain a rectangular domain: A planar parameterization of the point cloud must be obtained to fit tensor product B-spline surfaces. If the parametric domain is not rectangular, the MLS approximation is used to fill in data to make the domain rectangular.

4. Fitting: The final step involves fitting a tensor product B-spline surface to the point cloud and finding the trim curves.

Each of the steps is presented in detail in the following sections.

Further, this thesis performs an analytical study of several aspects of the above process, such as

1. A comparison of the effect the hole filling step on the fit with finding a minimum norm least squares solution using SVD without filling holes.

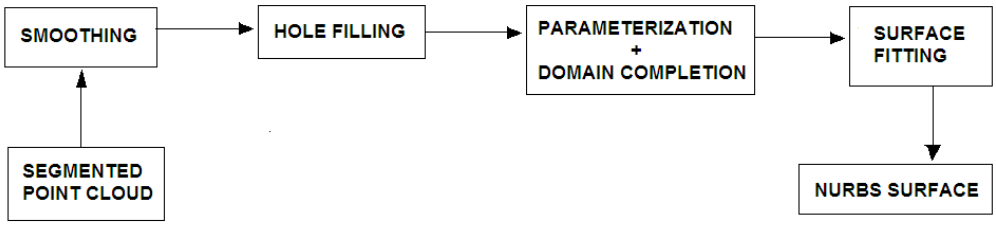


Figure 3.1. The framework

2. A comparison of blending of local fits(the approach proposed herein) during the fitting phase with a global least squares fit in terms of quality of fit, stability and computation speed.

3. The weighting function and the optimal size of neighborhood to use in a local weighted least squares fit during the hole filling phase and the fitting phase.

4. Quantifying the quality of fit obtained in the fitting phase.

CHAPTER 4

SMOOTHING

In order to deal with the noise in the input, this thesis introduces a smoothing step that removes the noise and outliers in the data. Smoothing is done by projecting each point in the point set onto the MLS surface calculated at that point as described in 2.1.3. Figure 4.1 shows a part of a noisy gear cover that has been smoothed using this method. During this stage, the normal at each point can also be computed (however care must be taken to get the right orientation of the normal by using additional information such as the underlying triangulation).

4.1 Boundary Smoothing

However, the method outlined above smooths the data in the local normal direction obtained through the MLS method. As a result, the boundary curve may contain significant noise even after smoothing the surface. That is, normals of the points in the boundary curve may not lie along the surface normals at these points. Hence, normal smoothing of the surface alone does not suffice.

An example of such a situation is shown in Figure 4.2 where the surface is effectively smoothed using the MLS projection though the boundary of the surface still contains significant noise. Figure 4.3 shows another such example.

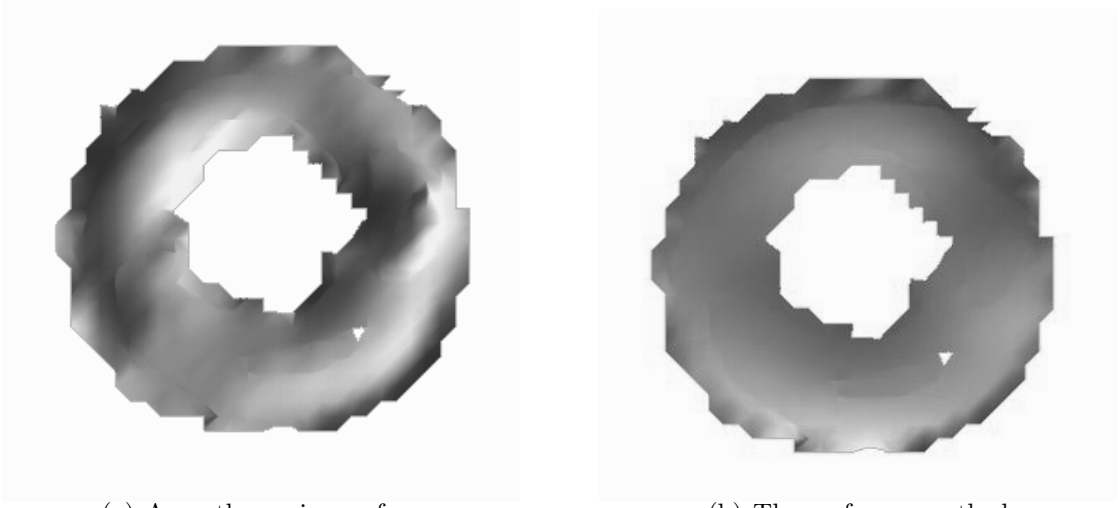
In order to address this situation, we introduce an additional step of smoothing the boundary by projecting each point in the boundary onto the MLS curve computed locally at the point [54]. The process of MLS projection for curves is described in the next section.

4.2 MLS Projection for Curves

In section 2.1.3 the MLS projection for surfaces, based on the pioneering work of Levin [7] has been described. This section aims at extending this procedure to curves.

(a) Noisy Data (b) The surface smoothed

Figure 4.1. An example of the smoothing process.



(a) A another noisy surface (b) The surface smoothed

Figure 4.2. An example of the tangential noise present in the boundary that is not eliminated by normal smoothing.

The MLS projection for surfaces proceeds by first finding a local neighborhood, then finding a local reference plane, then finding a local parameterization of the surface by projecting the points onto the plane, and finally fitting a quadratic surface and projecting the point onto this surface. Though we follow a similar procedure for curves, the extension for curves is not totally obvious since it involves finding a reference line instead of a plane and handling nonplanar curves.

(a) Noisy data

(b) Another example where tangential noise is present even after smoothing the surface using the MLS projection

Figure 4.3. An example of the tangential noise present in the boundary that is not eliminated by normal smoothing.

The method we have described for MLS projection for curves is similar to that for surfaces and has the following steps. For every point r on the curve,

1. Find a local reference line: Find a local neighborhood N_r consisting of N points. Let u be a unit vector in the direction of the optimal reference line and q be the projection of r on u as shown in Figure 4.4. For every point P_i belonging to N_r, $\sum_{i=1}^{N} ||(p_i - q) - \langle p_i - q, u\rangle\, u||^2 \theta(||p_i - q||)$ is minimized with respect to q and u. This is a nonlinear minimization process. Representing u using spherical coordinates we have $u = (\cos\gamma\cos\phi, \cos\gamma\sin\phi, \sin\gamma)$ for some γ and ϕ. This has two degrees of freedom, γ and ϕ. Making use of the fact that that the direction $q - u$ is perpendicular to u, there are two degrees of freedom for the point q. Hence there are four degrees of freedom during the minimization. We use the Powell minimization method to get the optimal values for q and u.

2. Now we have a local orthonormal basis, u, $(r - q)$ and $(u \times r - u)$. The neighborhood is transformed to this local orthonormal basis with q as the origin. For every point p_i, the new point in the local coordinate system become $q_i = (< p_i - q, u >, < p_i - q, r - q >, < p_i - q, u \times (r - q) >)$

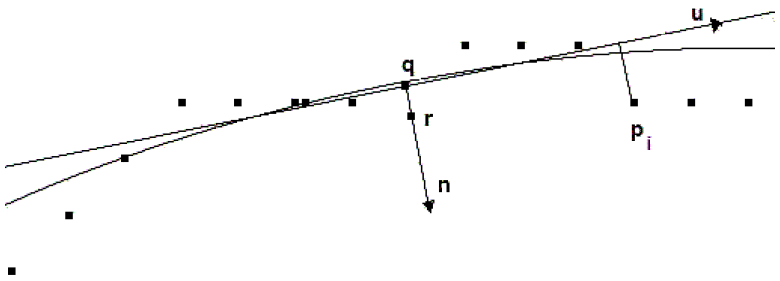


Figure 4.4. MLS projection procedure

3. Finally a quadratic curve is fit and the point is projected onto the curve. Since the curve might not be planar, we use a parametric quadratic curve $g(t) = (t, v(t), w(t))$.

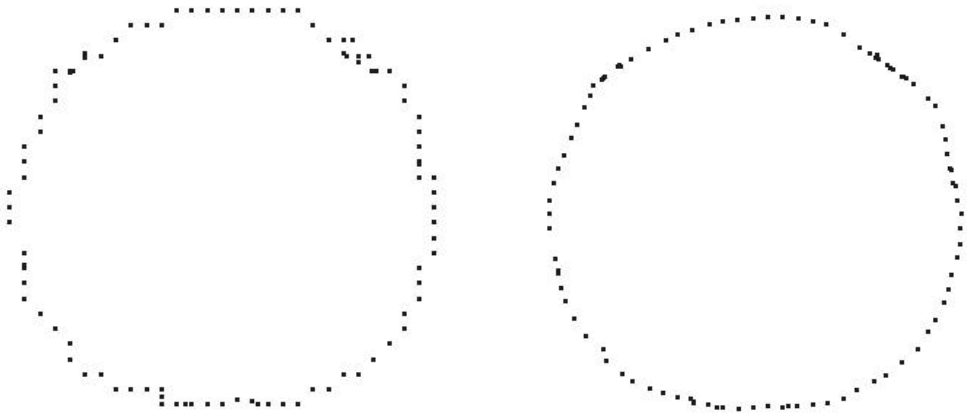
4. The curve evaluated at $t = 0$ using the moving least squares method gives the desired MLS projection.

The result of the 3D curve smoothing process is illustrated in Figure 4.5. The value of h plays an important role in this process throughout. Larger values of h result in smoother curves. However care must be taken that h is small compared to the smallest feature size. In Figure 4.6 a nonplanar space curve is smoothed using our technique. Also, Figure 4.6 shows different curves that result when smoothing is performed with different values of h.

4.3 Discussion

This method of smoothing works well for point clouds. However, when using this method to smooth triangular meshes, problems can arise because of the already existing triangulation. Some of the triangles might overlap with other existing triangles. The number of overlaps depends on the amount of noise and the kind of noise present in the input. For instance, when the input has normal noise alone, the probability of such overlaps lessens. These overlaps do not really affect the fitting process much though they can lead to some inconsistencies in the parameterization.

Although, very few overlaps were encountered with the input that was used for testing, this problem can be eliminated by re-triangulating the data after the



(a) Noisy Curve (b) Smooth Curve

Figure 4.5. An example of the curve smoothing process.

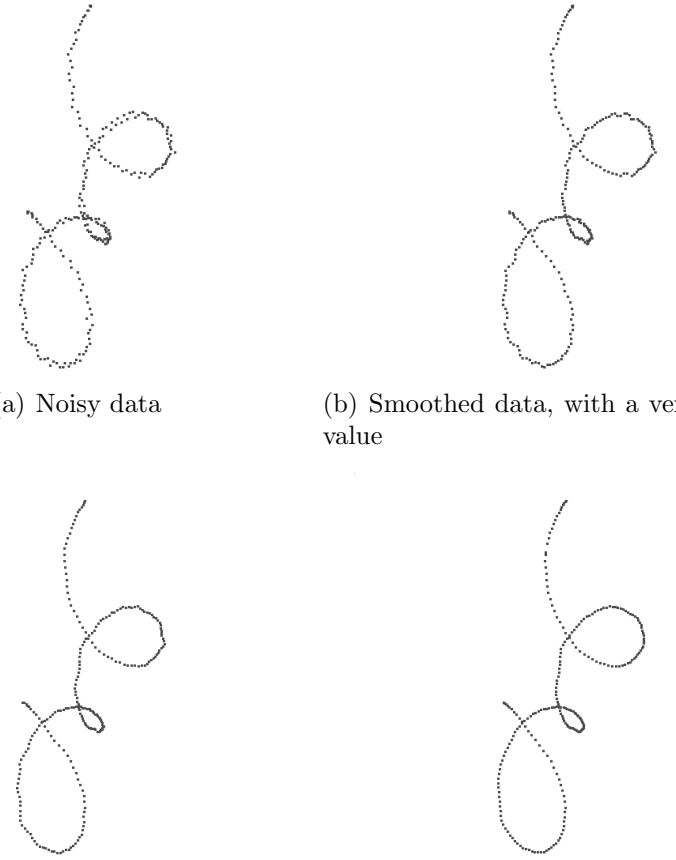
smoothing phase, to eliminate the inconsistencies. Several established triangulation techniques exist to triangulate the smoothed data. This has not been incorporated in this thesis due to time constraints.

Another way to deal with this problem would be to locally detect an inconsistency caused due to a modified vertex, because of smoothing, and do a local retriangulation. This is more efficient than a global retriangulation, since only a small fraction of vertices will actually cause flips or inconsistencies and retriangulation must be done only in these cases.

In Figure 4.7, the boundary of the wheel model is smoothed separately by this method. Figure 4.8 shows another example of the same.

4.3.1 Handling Sharp Corners in the Boundaries

Smoothing objects using the MLS projection suffers from the fact that it cannot preserve sharp features in the input. This does not become a major hindrance in the surface smoothing process, as we assume that we do not have any sharp features in the input since the data that we get is a result of a segmentation process. But this might pose difficulties during boundary smoothing. This is because, the boundary might still contain sharp features, though the interior of the surface might not have



(a) Noisy data

(b) Smoothed data, with a very small h value

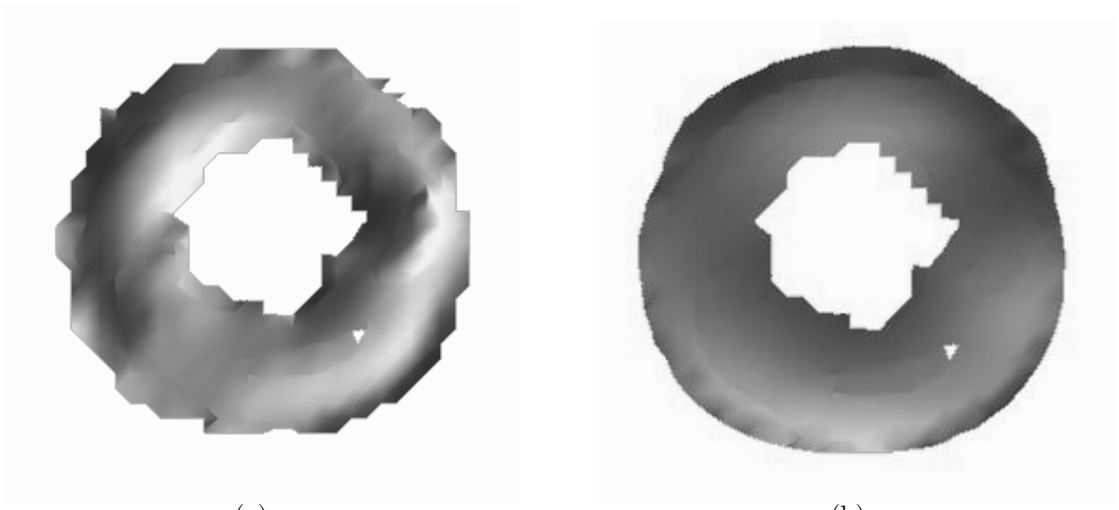
(c) Smoothed data, with a larger h value

(d) Smoothed data, with an even larger h value

Figure 4.6. The 3D curve smoothing process illustrated on a nonplanar spiral

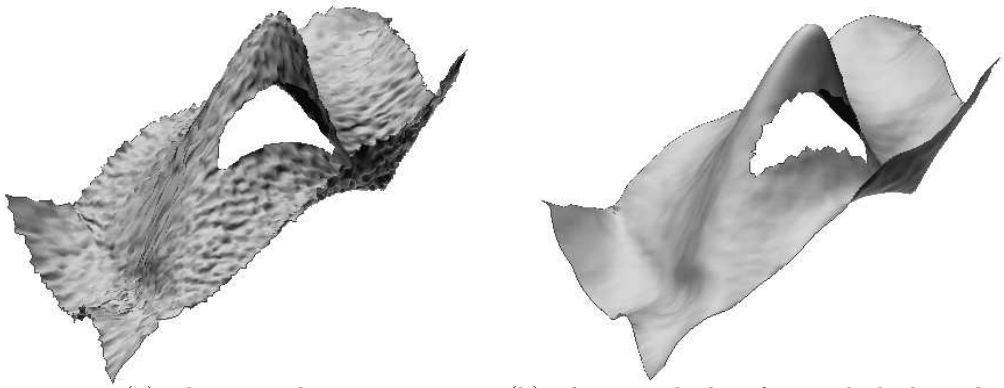
any. For instance, if we have a rectangular part of the segmented object, the corners of the rectangle will be smoothed.

To avoid this problem, we allow user intervention, to point out the sharp corners in the boundary. Now the boundary smoothing process is carried out for each individual piece of the boundary as described before and sharp features in the boundary are preserved. Figure 4.9 shows the process of smoothing a curve with sharp corners.



(a) (b)

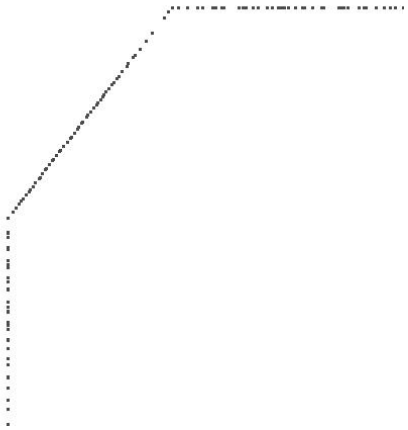
Figure 4.7. The wheel model after smoothing both the surface and the boundary.



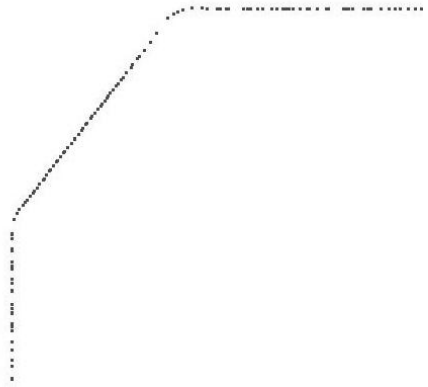
(a) The noisy data

(b) The smoothed surface with the boundaries smoothed using the curve smoothing process

Figure 4.8. The curve smoothing process illustrated to smooth the boundary of the surface to remove tangential noise



(a) A part of a curve containing sharp corners



(b) The sharp corners smoothed by the process of smoothing

Figure 4.9. A curve with sharp features that are not preserved when the entire curve is treated as a single piece.

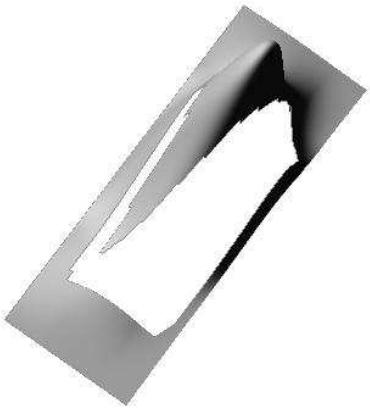
CHAPTER 5

HOLE FILLING

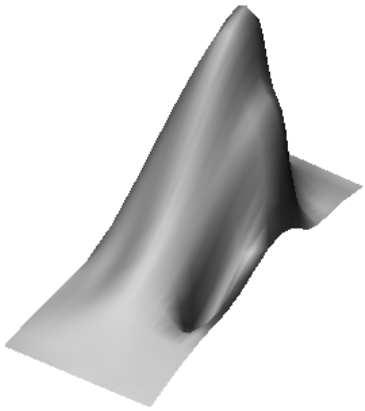
Holes in the data need to be filled for several reasons as discussed in Chapter 3. Filled models are not only visually appealing but are also easy to parameterize and can be fit without problems of rank deficiency. Figure 5.1 illustrates the effect of hole filling on fitting. Figure 5.1(a) shows a surface with a hole. Figure 5.1(b) shows the resulting surface when the fitting is performed using singular value decomposition without filling the hole. Figure 5.1(c) shows the surface obtained when the hole is filled. As observed in this figure, there is signifcant distortion when the hole is not filled.

We present a new hole-filling procedure to fill holes in triangular meshes based on the MLS projection procedure [55]. This method operates only in the vicinity of the holes. Hence it is efficient in terms of computation time. Apart from the role it plays in this framework, this method can be used with any surface reconstruction algorithm to fill holes as an independent postprocess and is easy to implement. At this stage in the framework, since we operate on triangular meshes for the basic data type, it is straightforward to find the boundaries of the hole by finding a connected set of edges that are associated with only a single triangle. This hole-filling algorithm can be extended to deal with holes in point clouds.

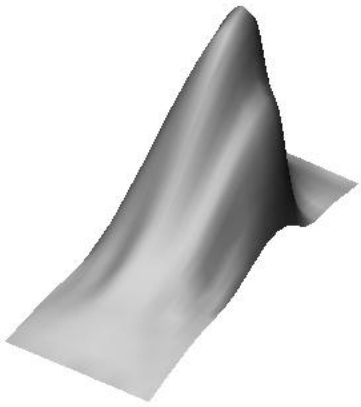
Since a global parameterization of the data is not available, the filling procedure needs to be local in nature to handle holes of arbitrary geometry. The algorithm consists of two alternating steps, the *make-convex* step and the *add-vertices* step and proceeds until the boundary size reduces to three vertices and the hole can be filled by inserting a single triangle. The *make-convex* step involves making a new triangle by adding an edge between adjacent pairs of edges in the boundary that make an angle greater than ϕ. The *add-vertices* step involves adding new vertices along the



(a) Data with a hole

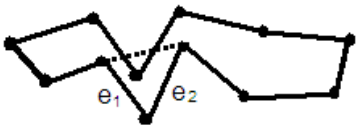


(b) The resulting surface when fitting is performed without filling the hole using singular value decomposition to obtain the least norm solution

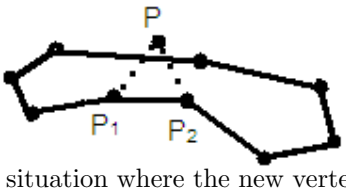


(c) The resultant fit obtained when the fitting is done after filling the hole

Figure 5.1. The effect of holes in the fitting phase



(a) A situation where the new edge introduced in the *make-convex* step crosses other neighboring boundary edges in the parametric domain



(b) A situation where the new vertex introduced in the *add-vertices* step crosses other neighboring boundary edges in the parametric domain

Figure 5.2. The situations that can arise if the various checks are not performed

boundary of the hole closing the hole radially inwards at each edge. This section discusses these two steps in more detail.

5.1 Steps in the Hole-Filling process

5.1.1 Make-Convex

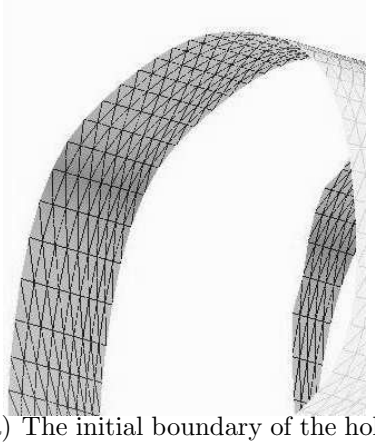
For every pair of adjacent edges b_1 and b_2 on the boundary, such that the angle between the edges is less than ϕ

1. A local neighborhood of the mid-point of the edge is obtained. A local reference plane is computed with respect to the mid point of the edge (as presented in [7] and described in section 2.1.3) .

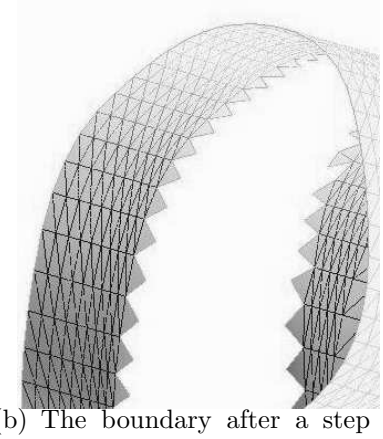
2. A local parameterization is obtained by orthographically projecting the local neighborhood onto the plane.

3. If a new edge joining b_1 and b_2 in the parametric domain does not intersect any other boundary edge in the parametric domain that lies within the neighborhood, the new edge is added to the mesh. A situation where this condition is violated is shown in Figure 5.2(a).

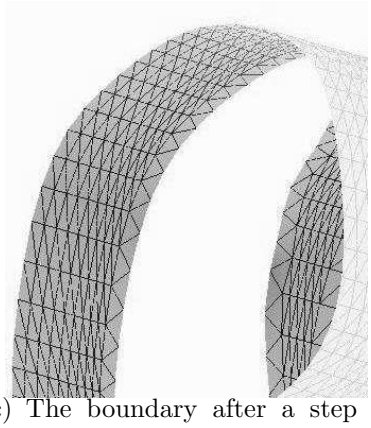
Since new boundary edges are introduced in this process, multiple passes of this step are made until no adjacent boundary edges make an angle greater than ϕ. The results of an intermediate stage of application of this process is shown in Figure 5.3.



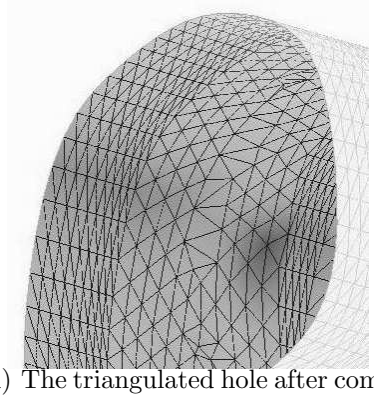
(a) The initial boundary of the hole



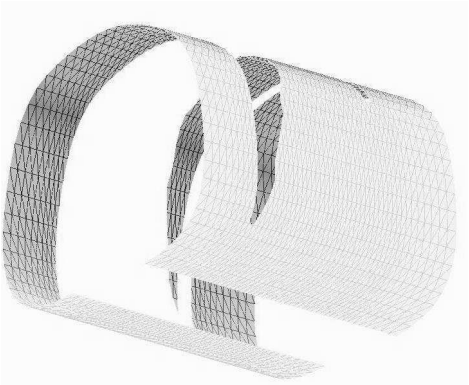
(b) The boundary after a step of *make-convex* and *add-points*



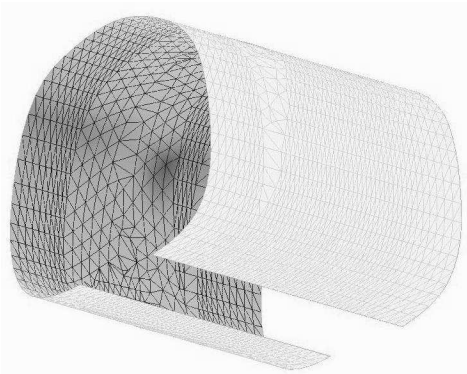
(c) The boundary after a step of *make-convex*



(d) The triangulated hole after completion of the hole-filling process



(e) A complete view of the surface with the hole



(f) A complete view of the filled surface

Figure 5.3. The various steps in the hole filling process.

5.1.2 Add-Vertices

1. For every edge e in the current boundary

   (a) A local neighborhood of the mid point of the edge is obtained. A local reference plane is computed as outlined in section 2.1.3 with respect to the mid point of the edge e.

   (b) A local parameterization is obtained by orthographically projecting the local neigbourhood onto the plane.

   (c) A new point along the perpendicular bisector of e in the local parameterization is chosen, at a specified distance d_e.

   (d) A new point on the surface is computed using a local MLS approximation.

2. For every new point p, the closest edge in the current boundary is found. Let the end points of the edge be p_1 and p_2. In order to ensure that well behaved triangles are obtained, a check is made to see if introducing a new triangle t with p, p_1 and p_2 crosses any other boundary edge in the local parametric domain. If so, the point p is discarded. Such a case is shown in Figure 5.2(b). If not, the triangle t is introduced. This process is shown in Figure 5.3(c).

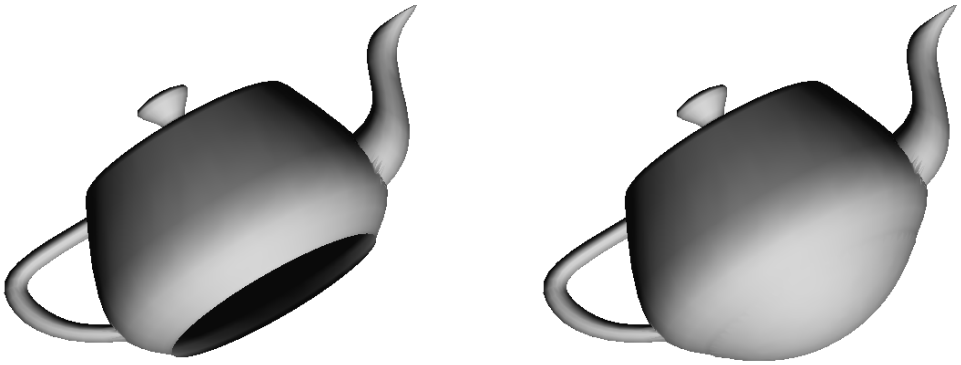
If no new vertices or edges have been introduced in the boundary after an iteration of *make-convex* followed by *add-vertices*, and the boundary size is greater than three vertices, the process is repeated with a reduced d_e and ϕ to promote progress.

The choice of d_e and the value of ϕ play a significant role in determining the shape of the triangulation. If ϕ is too large, ill-shaped triangles result that are elongated in one direction. We have obtained reasonable results for ϕ of about $5\pi/9$. Some alternatives for choosing d_e are to select a d_e so that the resulting triangle is equilateral in the parametric domain with side e, where e is the side under consideration, or to create an isosceles triangle in the parametric domain with two of the sides as the average edge length a. We choose the later method and choose d_e as $\sqrt{(4a^2 - e^2)/2}$.

5.2 Results

This section presents some examples of the hole filling process. Figure 5.4(a) shows the Utah teapot with a hole at the bottom. In Figure 5.4(b) this hole is smoothly

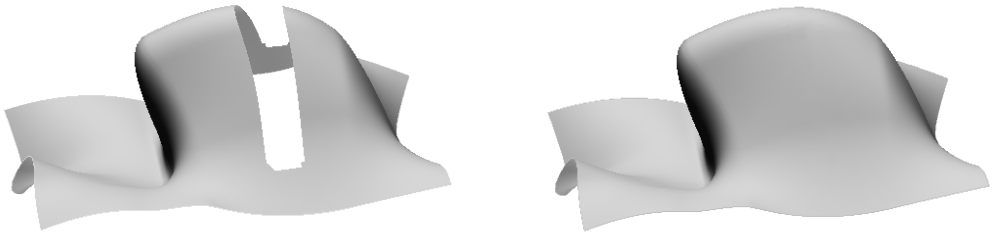
filled using this method. In Figure 5.5(a) a nonplanar hole is shown which cannot be parameterized by projection. Also mesh based methods that attempt to find the best fit plane in the vicinity of the hole using a PCA analysis tend to produce a local plane that lies along the direction of the hole. Parameterizing the vicinity of the hole by projecting onto this plane does not result in a single valued mapping. In Figure 5.5(b) this hole is filled using this method. Because this method fits a smooth surface in the vicinity of the boundary and interpolates to find new points it ensures that there are no visual discontinuities at the boundary. Figures 5.6(a) and 5.7(a) show two more surfaces with holes and 5.6(b) and 5.7(b) show the surfaces filled with this algorithm.



(a) Teapot model with a hole at the bottom

(b) Teapot model with the hole smoothly filled

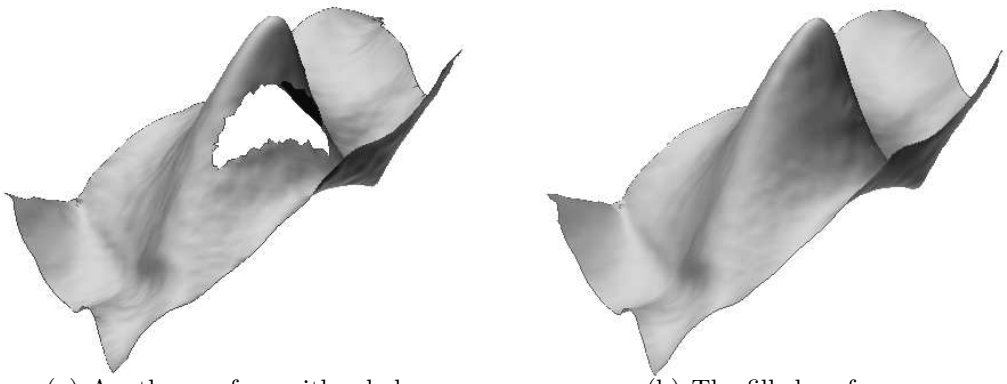
Figure 5.4. The bottom of the teapot model smoothly filled



(a) A surface with a hole whose vicinity cannot be parameterized by projection

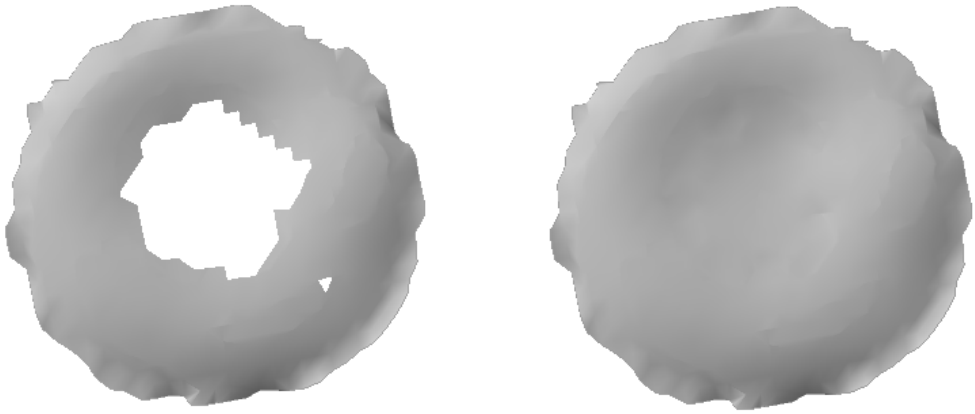
(b) The nonplanar hole filled using our method

Figure 5.5. Hole filling carried out for a hole whose vicinity is nonplanar



(a) Another surface with a hole (b) The filled surface

Figure 5.6. Another example of the hole filling process



(a) A hole in a wheel scanned from a tank model (b) The hole filled

Figure 5.7. Hole filling for the wheel model

CHAPTER 6

PARAMETERIZATION

This thesis creates an initial parameterization of the data by using a convex combination map. The mesh structure available with the input is utilized and mesh parameterization techniques have been employed to get a better parameterization. Floater's mean-value coordinates [48] are used as the parameterization technique in this thesis. In the occasion when triangulation information is not available, Floater's meshless parameterization [52] can be used.

The mean-value coordinates [48] method is a discretization of harmonic maps that are based on the fact that harmonic maps satisfy the mean value theorem. The method proceeds by mapping the boundary of the mesh to a convex polygon and solving a linear system of equations that express every point as a convex combination of its neighbors in the parametric domain. The weights used between a point v_i and v_j in the above process can be obtained as $w_{ij} = \tan(\delta_{ij})/2 + \tan(\gamma_{ij}/2)/||v_j - v_i||$ (as shown in Figure 6.1).

An important issue that arises while using convex combination maps is the method used to fix the parameter values for the boundary of the data. One option is to project the boundary onto a plane. However the projection must be a convex polygon in order to ensure that the mapping is a bijection. Also projecting the parameterization onto a plane does not guarantee that there are no flips or overlaps. In other words, the problem of finding a suitable plane for projection arises and it is not always guaranteed that one exists.

Another strategy is to directly map the boundary to a square (or a rectangle). This can be done by chord length once the first vertex of the boundary is fixed to map to a point on the rectangle. This does not always produce an intuitive parameterization. For nearly rectangular surfaces, a more intuitive parameterization can be obtained when selected points on the boundary are constrained to map to selected points on

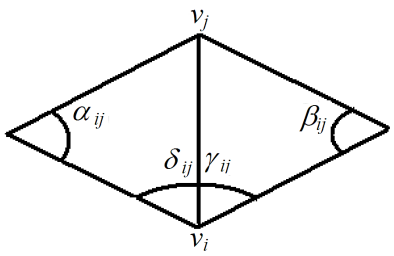


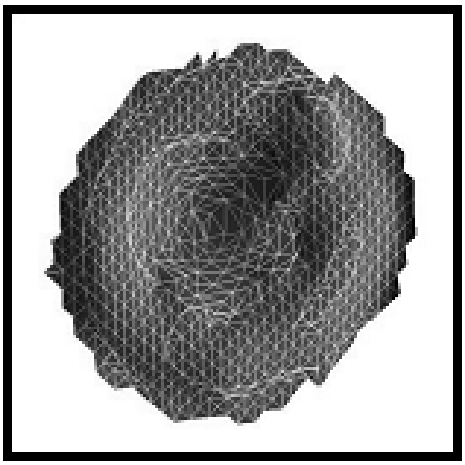
Figure 6.1. Mean-value coordinates

the rectangle, and the remaining points mapped to lie in-between these points by chord length.

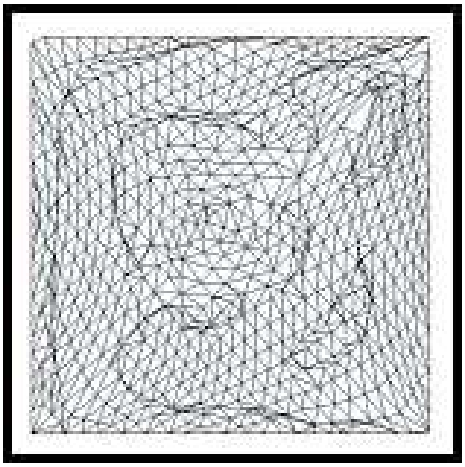
However, in the case of surfaces that do not naturally resemble a rectangle, Identifying four points on the boundary and mapping them to a rectangle might not suffice. The resulting parameterization, though rectangular, is unintuitive. In such an instance, the sharp corners in the boundary are identified and the boundary is mapped to a convex polygon whose sides are proportional to the curve lengths between these corners. In order to obtain a rectangular domain, a domain completion process, described in the next section is performed.

Hence, the approach we take is to use different strategies depending on the nature of the surface. If the projection of the boundary on the PCA plane has no crossovers and is convex, the projection can be used as the parameterization of the boundary. Else (if the projection of the boundary is self intersecting or nonconvex), an alternative strategy of specifying the corners and mapping them to the corners of a convex polygon, can be used.

Below are some results of parameterizing using harmonic maps by fixing the boundary to a rectangle using the chord length. This approach works in the case of the wheel model as shown in Figure 6.2, since the object under consideration has a more or less circular boundary with no sharp corners in the geometry. However, if we apply the same technique to an object with sharp corners, we might have a problem if we use a plain chord length mapping. In the second example of the bean model (Figure 6.3), the boundary parameterization is obtained by fixing the four



(a) A Wheel model

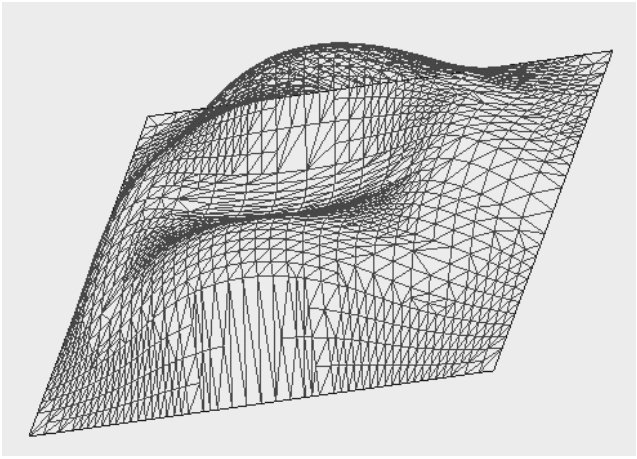


(b) Parameterization using mean-value coordinates

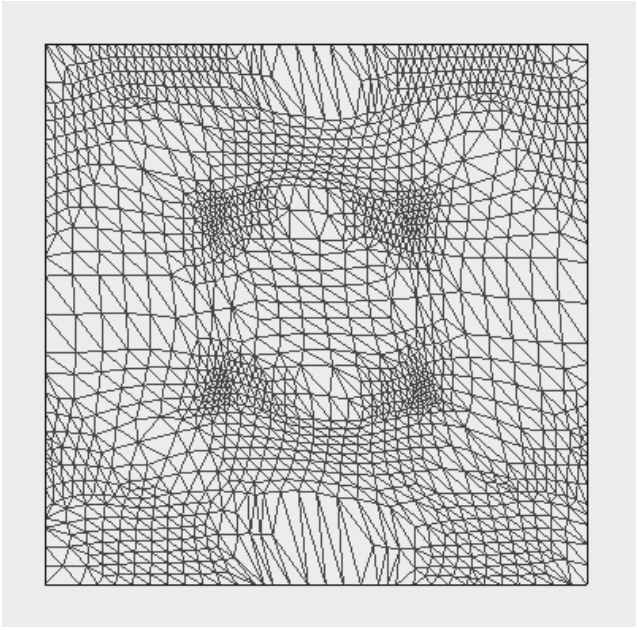
Figure 6.2. Wheel model parameterized using mean-value coordinates by fixing the boundary by chord length

boundary points to the four corners of a square and parameterizing the interior of the boundary by chord length. Figure 6.4 shows the parameterization obtained when the bean model is parameterized directly by chord length without identifying the sharp corners.

The linear system that needs to be solved to obtain a parameterization by convex combination maps, is characterized by the fact that it is large and sparse. Solving it the traditional way takes significant computation time. Conjugate gradient methods

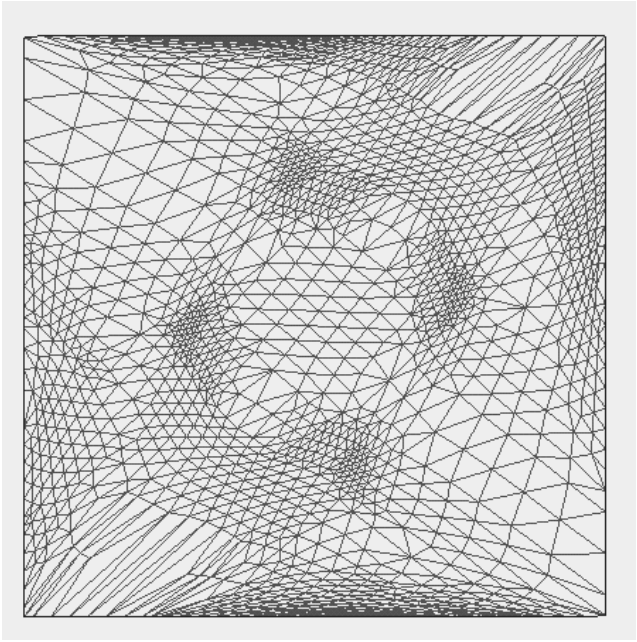


(a) The bean model

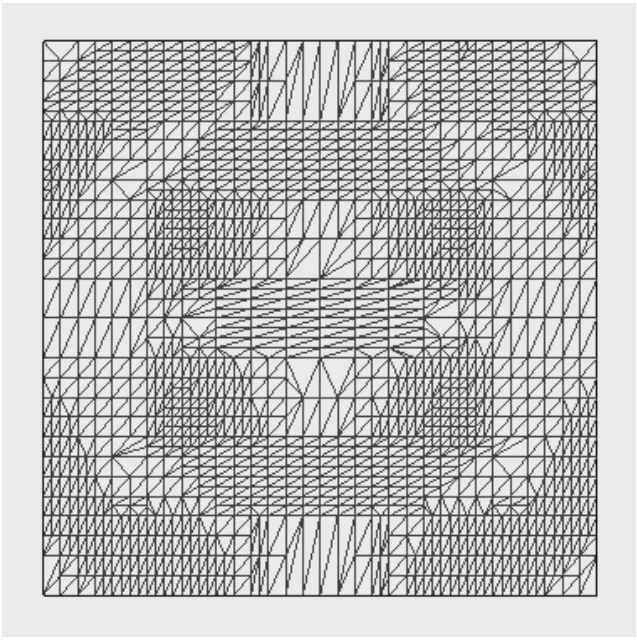


(b) Parameterization of the bean model

Figure 6.3. Bean model parameterized using the harmonic maps with four corner points fixed.



(a) Bean model parameterized by chord length without fixing the boundaries



(b) Bean model parameterized using the harmonic maps with four corner points fixed.

Figure 6.4. Bean model parametrized by chord length

can be used to best exploit the sparseness of the system. Hence, a bi-conjugate gradient sparse matrix solver has been used.

6.1 Domain Completion

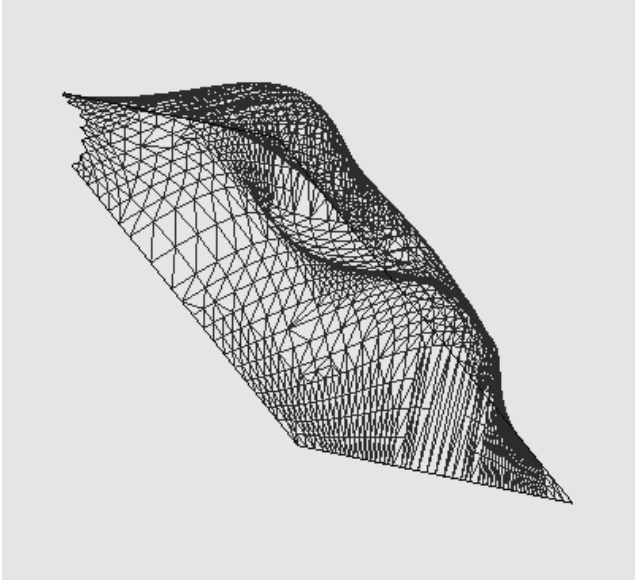
Tensor product NURBS surfaces require a rectangular parametric domain. However, the parameterization technique we use might not yield a rectangular domain as discussed earlier. To ensure that there is data throughout the chosen domain, and in order to make the domain rectangular this thesis proposes adding data to fill the domain. An added advantage with such an approach, apart from obtaining an intuitive parameterization, is that we can avoid distortion at the boundary during fitting, due to lack of data.

The parameterization of the boundary for the incomplete model can be obtained by mapping the sharp corners in the boundary to a convex polygon and parameterizing each piece of the boundary independently by chord length. This process is shown in Figure 6.5.

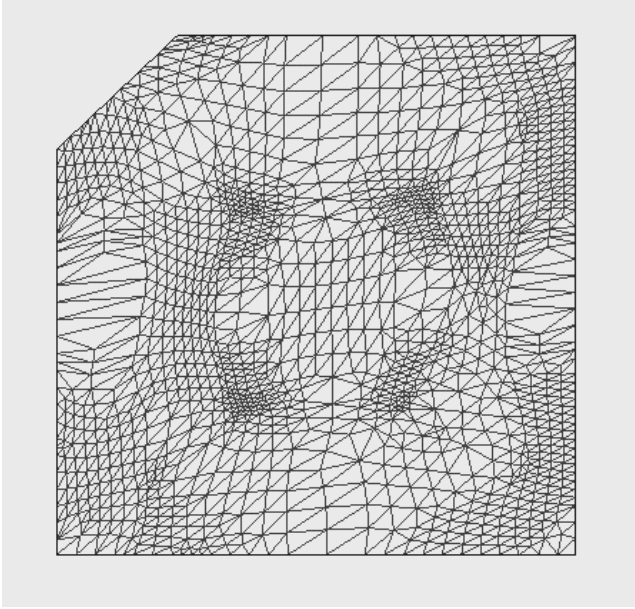
Using the local weighted least squares technique, new points can be inserted beyond the boundary, in a way similar to the hole filling step, by projecting each point onto a local MLS approximated surface. However, unlike the hole filling step, triangulation of the inserted points may not be necessary at this stage since they are not needed for the fitting step. (One purpose of the triangulation in the hole filling step is to aid the parameterization step.)

While this idea might not work equally well for all models, it produces desirable results for models that are rectangular in shape but are incomplete in the sense that a part of the rectangle is missing.

Figures 6.6(a) shows the bean model with a part missing. This part of the domain is filled using the domain completion process and the result is shown in the Figure 6.6(d). Figure 6.6(b) and 6.6(c) show the parametric domain in the process. Figure 6.7 shows the influence of the domain completion process on the resulting surface. Figure 6.8 shows another such example.

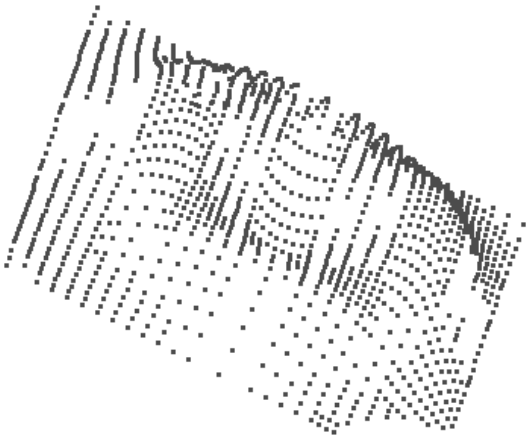


(a) Bean model with part of the boundary missing

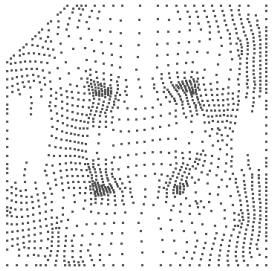


(b) The incomplete model parameterized using harmonic maps by specifying the five sharp corners

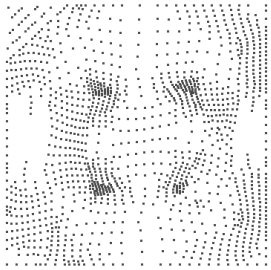
Figure 6.5. The domain completion process illustrated for the bean model



(a) An incomplete surface



(b) The parameterization of the incomplete surface obtained using convex-combination maps

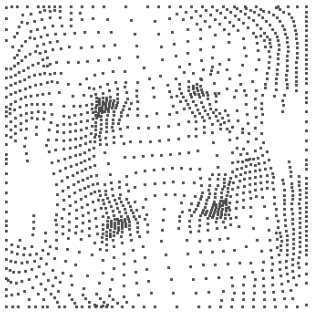


(c) The filled parametric domain

(d) The completed surface

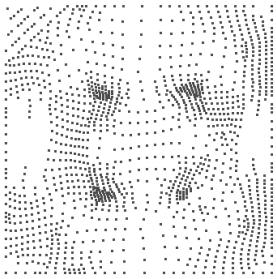
Figure 6.6. The domain completion process for the bean model

(a) The spline surface fit without the process of domain completion

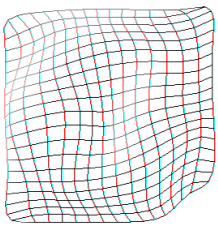


(b) The data parameterized by mapping the boundary of the incomplete model directly to a square.

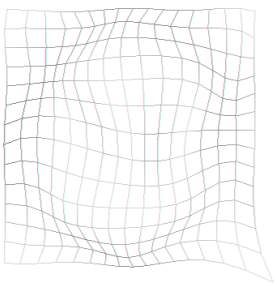
(c) The spline surface that results after the process of domain completion



(d) The parameterization obtained after domain completion

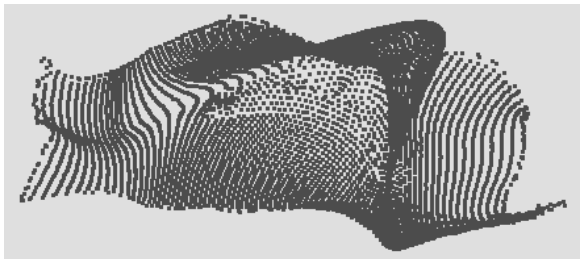


(e) The iso-parametric lines of the B-spline surface fit without the process of domain completion

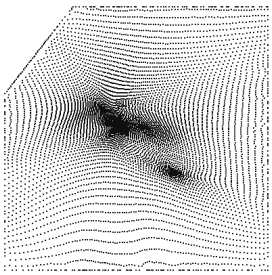


(f) The iso-parametric lines obtained through the process of fitting the surface after domain completion

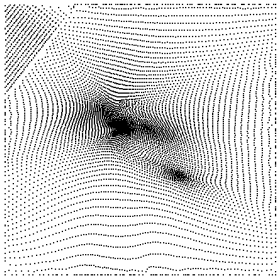
Figure 6.7. The influence of the domain completion process on the final surface



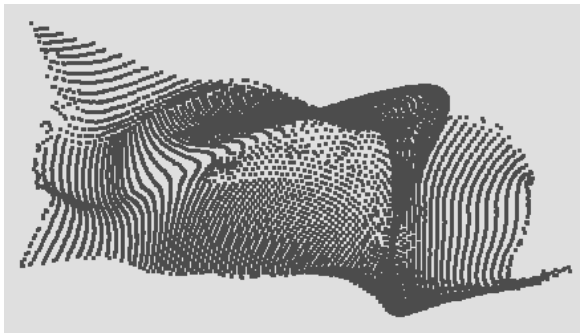
(a) An incomplete surface



(b) The parameterization of the incomplete surface obtained using convex-combination maps



(c) The filled parametric domain



(d) The completed surface

Figure 6.8. The process of filling the domain and adding new points for incomplete surfaces.

CHAPTER 7

FITTING

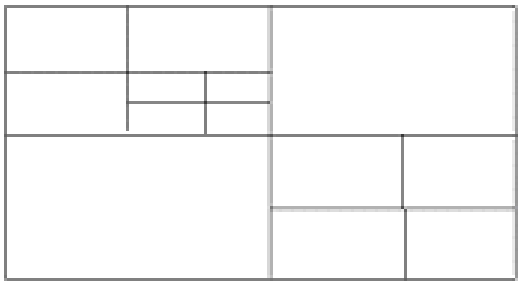
This chapter focuses on fitting B-spline surfaces to smoothed and filled data. The first step in this process is knot selection that is described in the next section.

7.1 Knot Placement

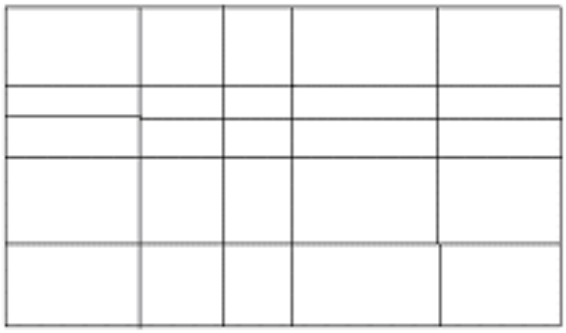
More detail can be captured in general by having a finer grid of control points. However, naively having a finer grid introduces control points even in places where they are not necessary and can lead to an unnecessarily large control grid and superfluous oscillations due to unnecessary degrees of freedom. So an important aspect of the fitting process is placing the adequate number of knots at the right locations.

The knot placement strategy used in this thesis, is to recursively subdivide the domain at the center of each region. The subdivision terminates when the data in the region can be fit locally with polynomial basis functions of degree 3, with a least squares error no more than κl, where l is the number of points in the patch, or until a specific recursion depth is reached, as shown in Figure 7.1, where the parameters κ and l can be specified by the user. When a patch cannot be subdivided anymore, the end points of the patch are added to the knot vector. The final knot vector is obtained by creating an open uniform knot vector from the result of domain decomposition by repeating the boundary knots.

Hence, the knot vector finally contains the end points of all the patches along with additional knots added at the corners to make it open uniform. If there are n_p patches in the u direction, there are $n_p + 1$ knots that are obtained from the end points of patches and hence there are $n_p + 7$ knots in all when additional knots are added to make it open cubic.



(a) The subdivision process



(b) Additional patches added in order to obtain a grid for making a knot vector for tensor product B-spline surface

Figure 7.1. Recursive domain decomposition

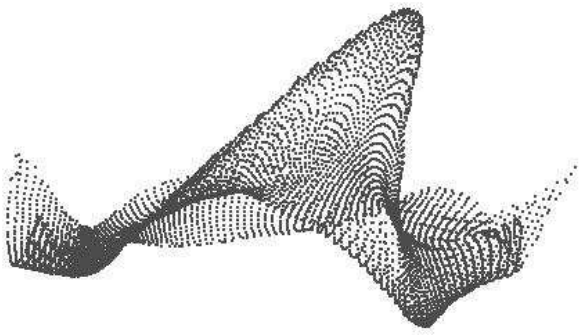
This domain decomposition technique gives relatively good results in reasonable time as shown in Figure 7.2.

While a more appropriate knot vector can be selected using free knot techniques discussed in section 2.5, this method is considerably fast compared to other free knot methods. Some ways to improve this method are suggested in the 9.3 section.

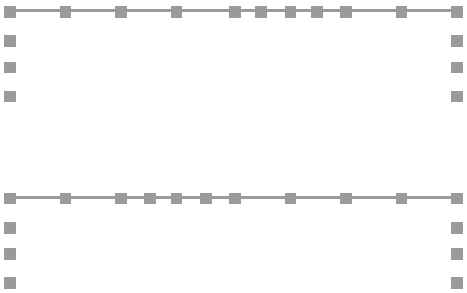
7.2 Fitting

7.2.1 Blending Local Fits

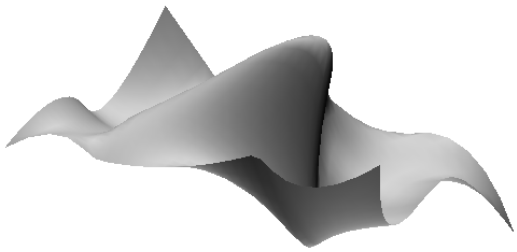
The fitting step produces a tensor product B-spline surface that represents the shape of the input, given the parameterization over a rectangular domain. Once the knot positions are determined, the task remains to determine the control points. In order to fit the surface, this thesis examines a new method that involves blending local



(a) The input point cloud



(b) Knot vector obtained using hierarchical domain decomposition



(c) The surface under consideration

Figure 7.2. The domain decomposition process

fits to obtain a global fit of the surface. This method is motivated by the fact that blending local fits can be faster than a global fit. This thesis further evaluates the proposed approach and compares it with a global least squares fit in terms of quality of the fit, speed and finally numerical stability (problems such as rank deficiency).

7.2.2 The Local Fit

To illustrate our method, this section discusses the blending local fits method to fit B-spline curves. Suppose the hierarchical domain decomposition process leads to n_p patches, the knot vector has a size of $n_p + 7$, with $n_p + 3$ basis functions. If the total number of data points is N, and we want to fit a B-spline curve $\gamma = (t, f(t))$

where $f(t) = \sum_{j=0}^{n_p+2} C_j \beta_{j,k}(t)$, given data points $\{(t_i, f_i)\}_{i=0}^{N-1}$, global least squares fit minimizes $\sum_{i=0}^{N-1} (\sum_{j=0}^{n_p+2} C_j \beta_{j,k}(t_i) - f_i)^2$

Let the n_p segments resulting from the domain decomposition process be $\{P_i\}_{i=0}^{n_p-1}$ and the mid-points of these patches be $\{M_i\}_{i=0}^{n_p-1}$ and the number of points in each patch be $\{N_i\}_{i=0}^{n_p-1}$.

For each patch P_p, the coefficients of the local B-spline fit, L_0^p, L_1^p, L_2^p and L_3^p are found by applying the least squares criterion to the points in each patch by minimizing $\sum_{i \in N_i} (\sum_{j=0}^{3} L_j^p \beta_{j+p,k}(t_i) - f_i)^2 w(t_i - M_p)$. where the function w is shown in Figure 7.3.

In other words, the four basis functions used for any patch are the four B-spline basis functions that are non-zero in that patch as shown in Figure 7.4.

The final curve is a blend of the local curve pieces that are fit independently, joined with C^2 continuity. We attempt constructing a global control mesh where control points of each patch coincide with the appropriate control points of the neighbouring patches by the process of averaging control points of neighbouring patches as described below.

If $\{G_i\}_{i=0}^{n_p+2}$ represent the control points of the global control mesh, then, $G_i = r_0 L_3^{i-3} + r_1 L_2^{i-2} + r_2 L_1^{i-1} + r_3 L_0^i$ where $(r_0 + r_1 + r_2 + r_3) = 1$. This process is illustrated in Figure 7.5.

The first and the last terms of the above expression have less significance compared to the terms at the center because the control point G_i has lesser effect on patch P_{i-3} compared to G_{i-1} and G_{i-2}. This is shown in Figure 7.6. Also, the weight given to points in the patch P_i is small while doing the local moving least squares fit for patch P_{i-1}, as discussed later.

Based on these considerations, several methods of blending the coefficients have been tried in this thesis, in the curve case, that are enumerated below.

1. $r_0 = r_1 = r_2 = r_3 = 1/4$. Hence, in this scheme, the coefficients of the global control mesh are obtained as $G_i = \frac{L_3^{i-3} + L_2^{i-2} + L_1^{i-1} + L_0^i}{4}$ for i from 3 to $n_p - 1$, $G_0 = L_0^0$, $G_1 = \frac{L_1^0 + L_0^1}{2}$, $G_2 = \frac{L_2^0 + L_1^1 + L_0^2}{3}$, $G_{n_p} = \frac{L_3^{n_p-3} + L_2^{n_p-2} + L_1^{n_p-1}}{3}$, $G_{n_p+1} = \frac{L_3^{n_p-2} + L_2^{n_p-1}}{2}$ and $G_{n_p+2} = L_3^{n_p-1}$.

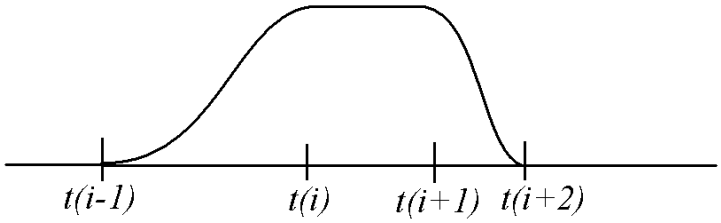
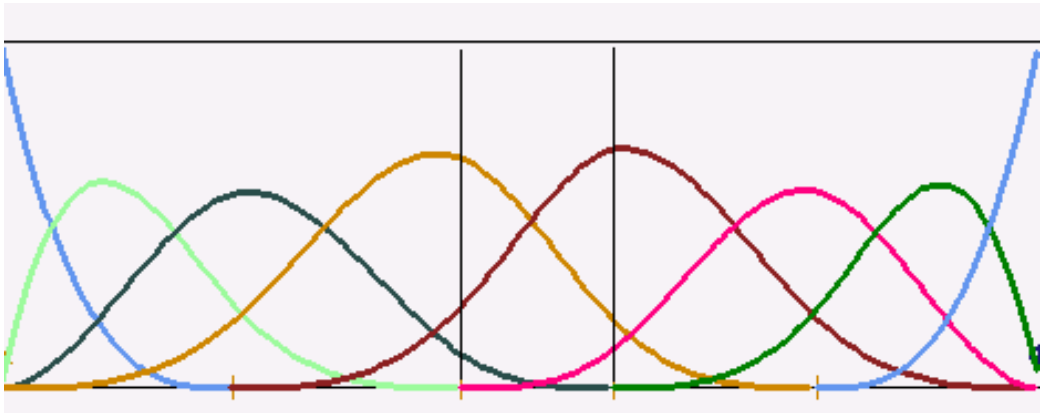
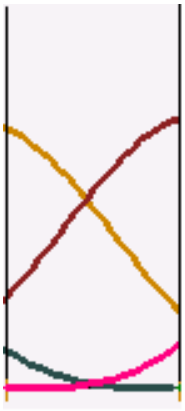


Figure 7.3. Weighting function that windows the neighborhood.



(a) Global basis functions corresponding to the knot vector



(b) Basis functions for a local fit in the particular interval

Figure 7.4. Basis functions

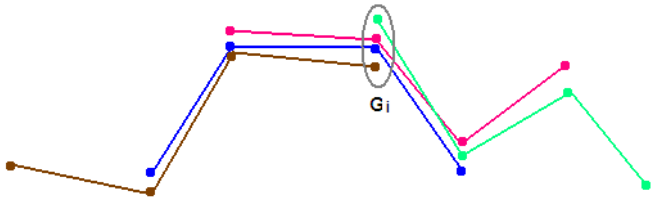


Figure 7.5. A figure showing the control polygons of four local patches and the process of blending

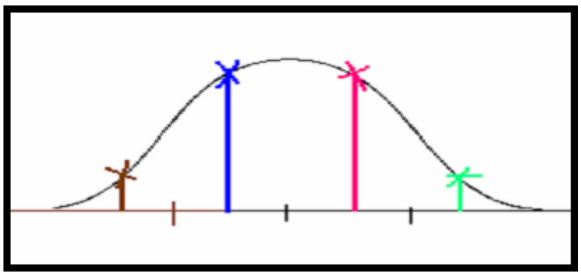


Figure 7.6. A figure showing a single B-spline basis function and its influence over each patch

2. $r_0 = r_3 = 0$ and $r_1 = r_2 = 1/2$.

   Hence, in this scheme, the coefficients of the global control mesh are obtained as $G_i = \frac{L_2^{i-2} + L_1^{i-1}}{2}$ for i from 2 to $n_p + 2$, $G_0 = L_0^0$, $G_1 = L_1^0$, $G_{n_p+1} = L_2^{n_p+1}$ and $G_{n_p+2} = L_3^{n_p+2}$. The coefficients of two adjacent segments are shown in the Figure 7.7.

3. $G_i = r_0 L_3^{i-3} + r_1 L_2^{i-2} + r_2 L_1^{i-1} + r_3 L_0^i$
   where $r_1 = r_2 = \frac{1}{s}\beta_{i,3}(t_i^*)$,
   $r_0 = \frac{1}{s}\beta_{i+1,3}(t_i^*)$ and
   $r_3 = \frac{1}{s}\beta_{i-1,3}(t_i^*)$
   where $s = \frac{\beta_{i+1,3}(t_i^*) + 2\beta_{i,3}(t_i^*) + \beta_{i-1,3}(t_i^*)}{4}$
   and $\{t_0^*, t_1^* ... t_i^*, t_{i+1}^*, ... t_{n_p+2}^*\}$ are the node values.

4. $G_i = r_0 L_3^{i-3} + r_1 L_2^{i-2} + r_2 L_1^{i-1} + r_3 L_0^i$
   where $r_1 = \frac{1}{s}\beta_{i,3}(\frac{t_{i+2} + t_{i+3}}{2})$,
   $r_2 = \frac{1}{s}\beta_{i,3}(\frac{t_{i+1} + t_{i+2}}{2})$,

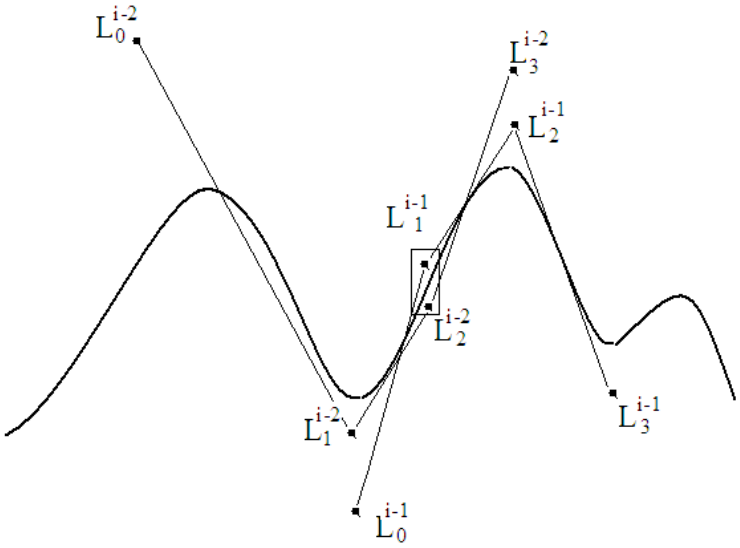


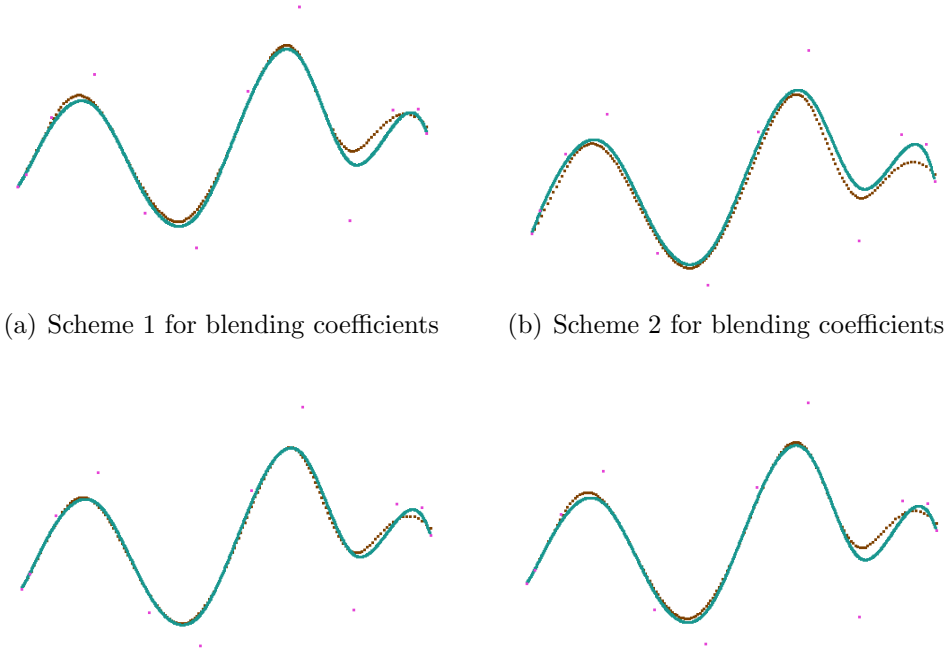
Figure 7.7. A figure showing the control polygons of two local patches and the process of blending

$r_0 = \frac{1}{s}\beta_{i,3}(\frac{t_{i+3}+t_{i+4}}{2})$ and
$r_3 = \frac{1}{s}\beta_{i,3}(\frac{t_i+t_{i+1}}{2})$
where $s = \frac{\beta_{i,3}(\frac{t_i+t_{i+1}}{2})+\beta_{i,3}(\frac{t_{i+1}+t_{i+2}}{2})+\beta_{i,3}(\frac{t_{i+2}+t_{i+3}}{2})+\beta_{i,3}(\frac{t_{i+3}+t_{i+4}}{2})}{4}$.

The results obtained by applying the scheme to the curve case is shown in Figure 7.8. As seen in the figure, the curve obtained by using scheme 3 has the least error and is the method used in this thesis.

The local fit for a surface patch, is done using local weighted least squares with respect to the mid point of each patch in the parametric domain, in a way similar to the curve case. The basis functions used for the local fit are tensor product cubic B-spline basis functions that are nonzero in the interval under consideration as shown in Figure 7.4. Since 16 basis functions are nonzero for a given knot interval, upon performing the local fit, 16 coefficients are obtained.

These are the 16 control points of a local patch. In order to obtain the control points that constitute the global control mesh, the control points of four adjoining patches corresponding to the same location in the parametric domain, are blended, in a method smilar to the curve case. This process is shown in Figure 7.9. At the boundaries, the control points of each pair of adjoining patches are blended.



(a) Scheme 1 for blending coefficients

(b) Scheme 2 for blending coefficients

(c) Scheme 3 for blending coefficients

(d) Scheme 4 for blending coefficients

Figure 7.8. Results obtained for the curve case with various blending schemes

The choice of weighting function to use in this process affects the result We use one that gives more weight to the interval under consideration and partial weight to the neighboring patches so that the local surface blends smoothly into the neighboring patches. With a knot vector $\{t_0,t_1...t_i,ti+1,\ ...t_n\}$, we use the the windowing function

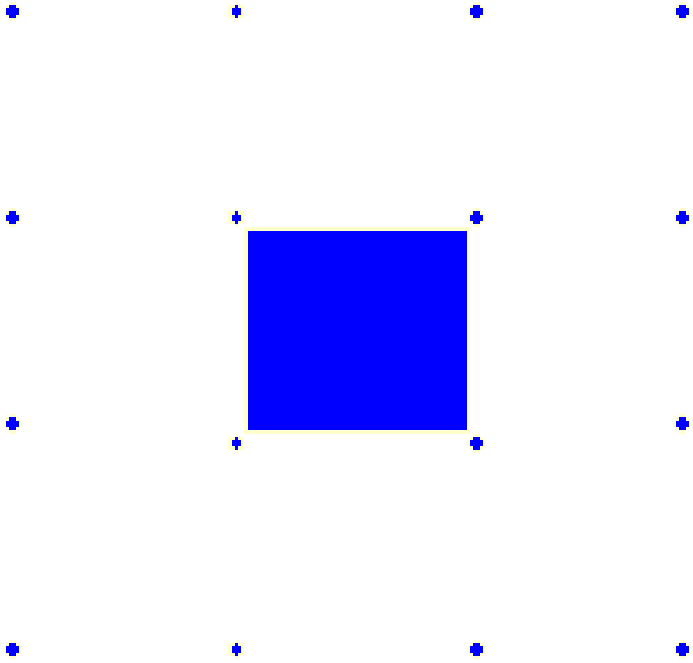
$$w(t) = \begin{cases} e^{\frac{(t-t_i)^2}{(t_i-t_{i-1})^2/4}} & t < t_i \\ 1 & t(i) \leq t \leq ti+1 \\ e^{\frac{(t-t_{i+1})^2}{(t_{i+1}-t_i)^2/4}} & t > t_{i+1} \end{cases}$$

Hence, we give full weight to the interval under consideration and decrease the weight exponentially beyond the interval as shown in Figure 7.3.

7.3 Quantifying the Quality of Fit

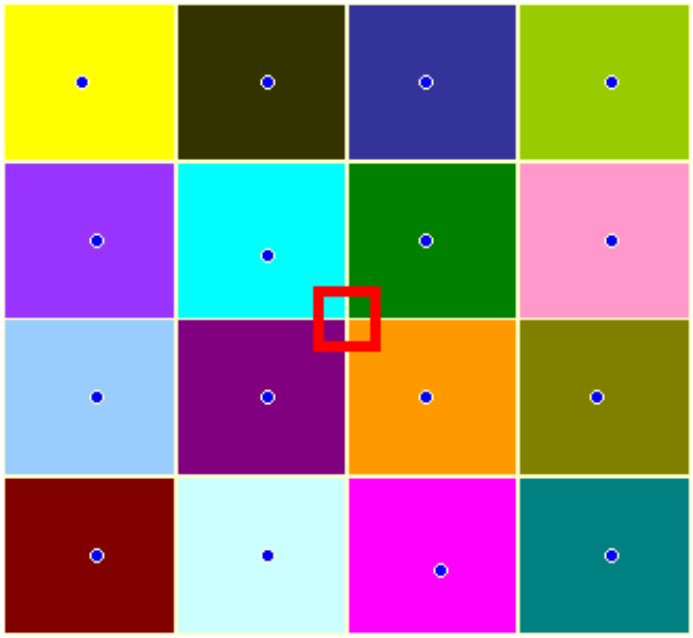
In this thesis, the quality of the fitting process is quantified in two ways using the parametric error and the minimum distance error.

Parametric error is the average distance between the data points and points on the surface evaluated at the parameter values corresponding to the data points. if



(a) The nodes corresponding to control points of a local fit for a uniform knot vector

(b)



(c) The control points of 16 adjacent patches being blended (for a uniform knot vector)

Figure 7.9. Control points when blending local fits

$S(u,v)$ is the fitted surface and there are N data points $\{P_i\}_{i=1}^{N}$ with parameter values $\{U_i\}_{i=1}^{N}$ where $U_i = (u_i, v_i)$, the parametric error can be expressed as

$\sum_{i=0}^{N} |P_i - S(U_i)|$

Minimum distance error is the average distance between the data points and the corresponding closest points on the surface corresponding to each data point. if $S(u,v)$ is the fitted surface and there are N data points $\{P_i\}_{i=1}^{N}$ with corresponding closest points $\{C_i\}_{i=1}^{N}$, the minimum distance error can be expressed as

$\sum_{i=0}^{N} |P_i - C(U_i)|$

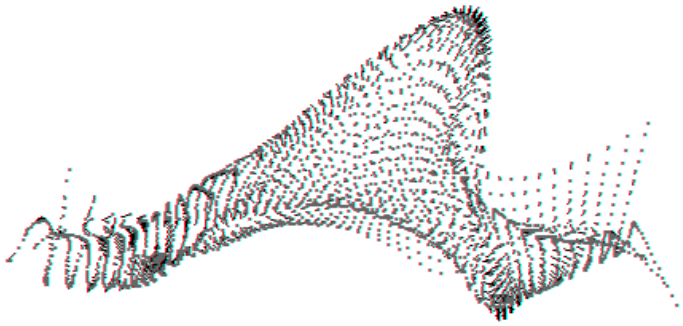
7.4 Analysis of Blending Local Fits

The preferred method for solving the linear least squares problem for data that is not well conditioned is the technique of singular value decomposition. This method takes a running time of the order $4\ell m^2 - 4m^3 + O(\ell^2)$ where there are ℓ data points and m control points to solve for.

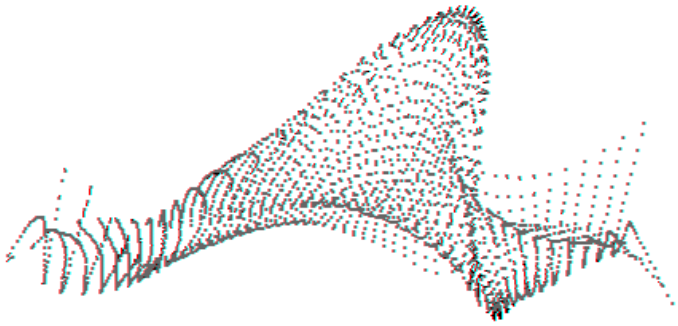
The blending local fits method provides substantial speed up computationally by using a constant $m = 16$ for each individual patch. Hence, if a is the number of rows and b is the number of columns in the global control mesh, the blending local fits method solves the normal equation for each patch in the control mesh. The total number of patches is $(a-3)*(b-3)$. So the number of points in each patch is roughly $\ell/((a-3)*(b-3))$.

The time taken to solve for the 16 control points in each patch is $O(\ell/((a-3)*(b-3))*16^2)$ or $O(\ell/((a-1)*(b-1)))$. Hence the total time taken is $O((a-3)*(b-3)*\ell/((a-3)(b-3)))$ or $O(l)$. In effect the blending local fits method reduces the runtime complexity of B-spline surface fitting from $O(\ell m^2)$ to $O(\ell)$. Hence the size of the control mesh does not have much effect on the complexity of the fitting algorithm. In other words, significant speedup can be realized for larger control meshes.

The quality of fit obtained using the blending local fits method largely depends on size of the neighborhood chosen and the weighting function used. Figure 7.10 shows the error obtained when approximating the data using the blending local fits method with a 13x13 control grid for 2500 points for the example shown in Figure 7.11. Figure 7.12 shows the error as a color map where red denotes the maximim error and green

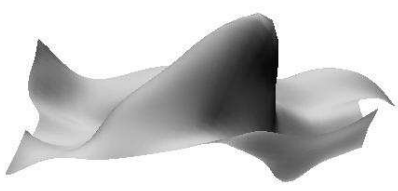


(a) Lines indicating the distance between points on the fitted surface and actual points for the same parameter value using the blending local fits method.

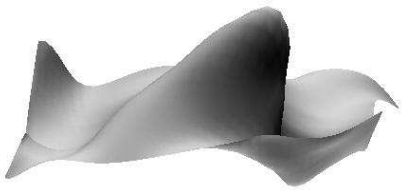


(b) Lines indicating the distance between points on the fitted surface and the actual points for the same parameter value using the global least squares fit method.

Figure 7.10. Comparison of the quality of fit of the global least squares fit method with the blending local fits method

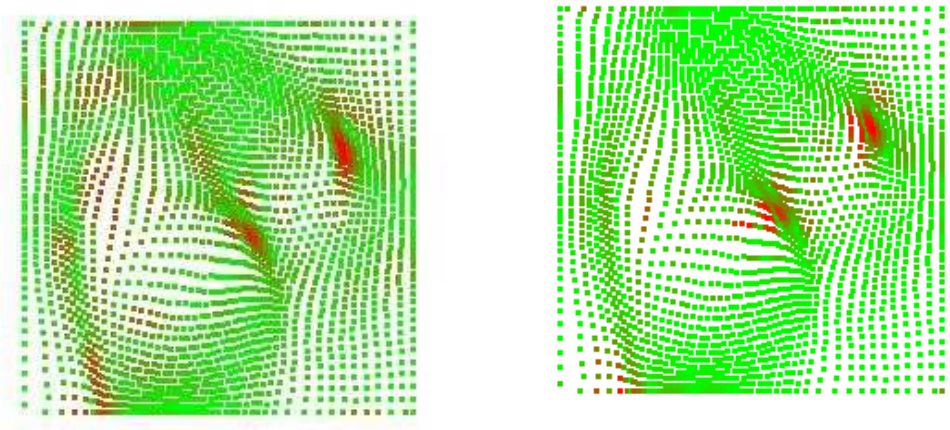


(a) The final fit surface with a local least squares method.



(b) The final fit surface with a global least squares method.

Figure 7.11. The final fit obtained when using the global least squares fit method and the blending local fits method



(a) Error with the blending local fits method. (b) Error with the global least squares fit method.

Figure 7.12. Comparison of the quality of fit of the global least squares fit method with the blending local fits method by showing the error as a color map (red denotes regions of high error and green indicates regions of error)

denotes minimum error. The actual error values obtained are tabulated in Table 7.1. The results obtained by using the process on the bean model are tabulated in Table 7.2.

Hence the blending local fits method does not out perform the global least squares fit method in terms of accuracy. However, there is a substantial benefit in terms of speed and the blending local fits method can be much faster than the global least squares fit method for large knot vectors. This is because the global least squares fit is particularly geared towards reducing the overall error while the blending local fits method attempts to reduce the error in each individual patch. In other words, the

Table 7.1. Comparison of the Blending Local Fits method carried out by blending coefficients using scheme 2 and scheme 3 with the global least squares fit for 2500 points with a 15 x 15 knot vector

Method	Minimum Distance Error	Parametric Error
BLF (Scheme 2)	0.1	0.163
BLF (Scheme 3)	0.073	0.11
GLSF	0.047	0.076

Table 7.2. Comparison of the Blending Local Fits (BLF) method (with scheme 2 of blending) with the Global Least Squares fit(GLSF) method in terms of accurcy and run time (in seconds) for 1458 points.

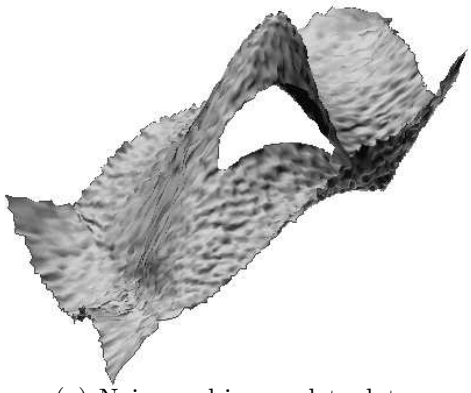
Method	Knot Vector	Minimum Distance Error	Parametric Error	Time
BLF	11x11	0.0015398	0.041739	9
GLSF	11x11	0.008580	0.025330	13
BLF	15x15	0.011727	0.024584	12
GLSF	15x15	0.004922	0.012207	42

global least error configuration that global least squares fit obtains might not be the least error configuration for each patch.

7.5 Results

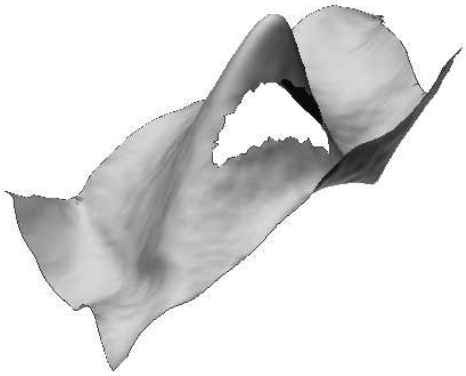
This section demonstrates the entire framework on noisy data shown in 7.13(a) with 8100 points. 7.13(b) shows the noise at the boundary that remains after normal smoothing of the surface. The noise in the boundary is removed by smoothing the boundary as shown in the Figure 7.13(c). The hole in the input is then filled using the hole filling algorithm shown in Section 5. The final fit, with a 13×13 control mesh, as shown in Figure 7.13(e) is obtained using the blending local fits method. The knot vector obtained using the adaptive domain decomposition technique is shown in Figure 7.13(f).

Figure 7.14 shows another example of the entire framework. The incomplete bean model is shown in Figure 7.14(a). Figure 7.14(b) shows this model filled. In Figures 7.14(c) and 7.14(d) the resulting surface and the parameterization after the process of domain completion are shown. In Figure 7.14(e) a B-spline fit for the bean model is shown, with the corresponding knot vector shown in Figure 7.14(f).

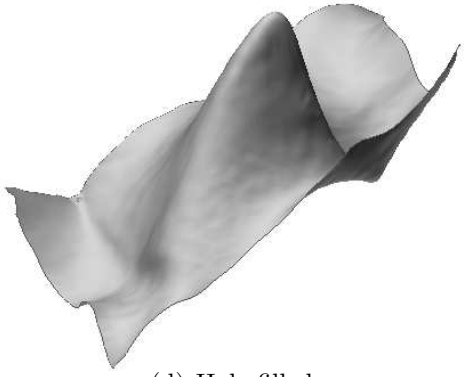


(a) Noisy and incomplete data

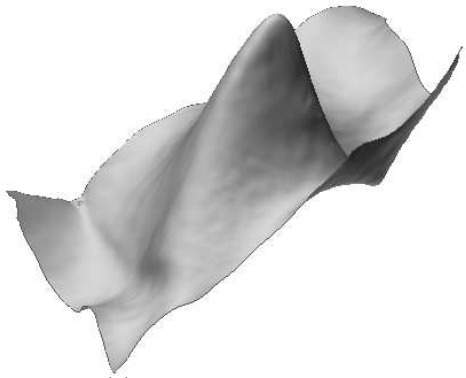
(b) Smoothed data, without the boundaries smoothed



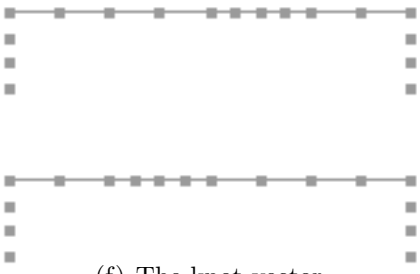
(c) The smoothed surface



(d) Hole filled

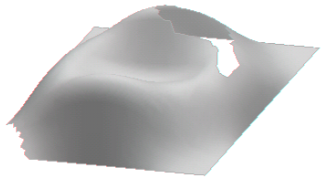


(e) B-spline surface fit

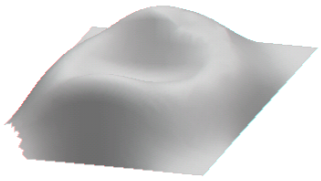


(f) The knot vector

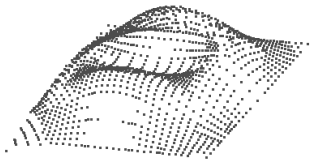
Figure 7.13. An example demonstrating the entire framework



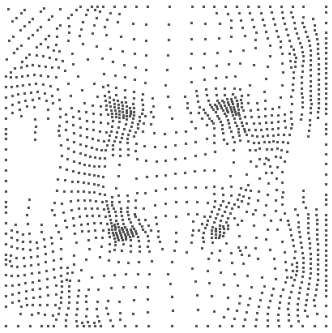
(a) Incomplete data



(b) The surface with the hole filled

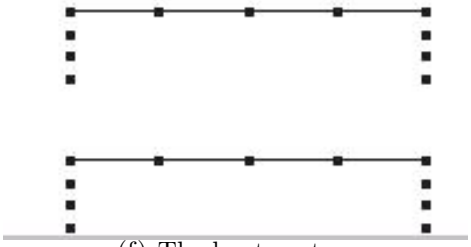


(c) The surface after the process of domain completion



(d) The parameterization after the process of domain completion

(e) B-spline surface fit



(f) The knot vector

Figure 7.14. The entire framework on the bean model

CHAPTER 8

CONCLUSION

This thesis presents a complete framework to convert point cloud data with moderate complexity into a tensor product B-spline surfaces to facilitate making trimmed NURBS or to obtain a tensor product B-spline surface for a patch of the original data set. This includes preprocessing steps such as smoothing, hole filling, parameterization, adding data at the boundaries when necessary and finally fitting the surface. Though several patch-based methods and softwares already exist to solve the reverse engineering problem, this method aims to deal with point clouds that have problems like noise, holes in the geometry and missing data. Also this method aims to capture more detail in a single patch by deciding where to place the knots using a hierarchical subdivision of the domain and uses a combination of local weighted least squares approximations to find the control points of the tensor product surface as a whole. Further, this thesis aims to provide an independent and easy to implement method (that is local in nature) to fill holes or missing data in a point cloud or a triangular mesh model efficiently.

CHAPTER 9

FUTURE

This chapter discusses possible research directions, to extend the work in this thesis.

9.1 Reparameterization

An idea under consideration, is the prospect of reparameterizing the data based on an already obtained parameterization, or the fitted surface. Reparameterization can be performed by stretch minimization for triangular meshes, and iterative reparameterization for parameter correction based on the closest point to the fitted surface from every data point.

9.2 Domain Specific Methods

Domain specific methods often yield good results compared to fully automated methods though they work for only a subset of the input. We have tried to exploit some domain knowledge, for instance in the case of smoothing the boundary curve of objects with a rectangular geometry and parameterizing objects with rectangular geometry. However, more use of domain knowledge can be made in many cases. For instance objects with circular geometry, can be parameterized onto a circle, using polar coordinates, with iso-parameteric lines being concentric circles and radial straight lines in the u and the v directions, respectively.

9.3 Knot Selection

Our knot selection process proceeds by dividing the domain at the center recursively until a specified tolerance is achieved. While this process facilitates an automated way to find knot vectors based on the complexity and the detail in the model it might not find the optimal knot vector and and may result in more knots than

required. While we have not addressed this problem in the thesis, due to constraints of time, there are broadly two ways to approach this situation. The first technique is to perform knot removal when the error after removal of the knot is less than certain predetermined constant (This indicates that the knot is actually not necessary.)

The second approach would be not to introduce such knots in the first place. One way to achieve this is to use a slightly different knot placement technique that decomposes the domain by dividing at the right position at each stage rather than divide it at the center. One way to approach this would be to find the place where there is maximum error and introduce the new knot at that place. Another way of choosing where to place the new knot is to use a nonlinear process, similar to the free-knot technique at each stage in the recursive process. While this coupled with the recursive domain decomposition technique is faster than choosing each knot using the free knot technique with the entire data, it takes significant time compared to simpler techniques described above.

9.4 Hole Filling for Point Clouds

The hole-filling algorithm described above, can be extended to use it without the requirement of a mesh structure. However, to have such an algorithm for point clouds, there must be a strategy to detect holes and there must be a way to chose the right location to insert the new point. Further, a similar strategy can be used to detect areas where there is a low sampling density and change the sampling.

9.5 Handling Lack of Data

Another way to handle a situation where there is lack of data can be to ignore some of the knots in this area and fit a coarser surface, without any knots positioned where there is a lack of data. Now knots can be introduced at the appropriate locations by refining the surface to get the actual surface.

9.6 Continuity between Patches

This thesis considers the process of fitting a surface to a single patch of a segmented model. However, in several situations, it is necessary to deal with an entire model.

Hence, the control points of each patch in the segmented model can be constrained so that adjacent patches join smoothly.

9.7 Meshless Method

Finally, the need of a mesh can be eliminated if the data can be parameterized without a mesh. One possible way of doing this is to parameterize data using the meshless method proposed in [52]. (We obtained better results by parameterizing using [48] and hence assume an underlying mesh structure.)

REFERENCES

[1] C. Clenshaw and J. Hayes, "Curve and surface fitting," *J.Inst Maths Applics*, 1965.

[2] P. Dierckx, "An improved algorithm for curve fitting with parametric functions," Tech. Rep. TW54, Department of Computer Science, Katholieke University Leuven, Belgium, 1981.

[3] D. L. B. Jupp, "Approximation to data by splines with free knots," *SIAM Journal on Numerical Analysis*.

[4] C. de Boor and J. R. Rice, "Least squares cubic spline approximation ii- variable knots," Tech. Rep. CSD TR 21, Purdue University, April, 1968.

[5] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, December, 1979.

[6] A. Marc, B. Johannes, C.-O. Daniel, S. Fleishman, D. Levin, and T. S. Claudio, "Computing and rendering point set surfaces," *IEEE Transactions on Computer Graphics and Visualization*.

[7] D. Levin, "Mesh-independent surface interpolation," *Geometric Modeling for Scientific Visualization*.

[8] J. Hoschek and D. Lasser, *The Fundamentals of Computer Aided Geometrical Design*. A K Peters, 1993.

[9] M. Plass and M. Stone, "Curve-fitting with piecewise parametric cubics," *Proceedings of ACM SIGGRAPH*.

[10] C. L. O'Dell, "Approximating data with parametric b-splines for computer aided design," *Masters Thesis, University of Utah*.

[11] G. Greiner and K. Hormann, "Interpolating and approximating scattered 3d data with hierarchical tensor product splines," *Surface Fitting and Multiresolution Methods, A. Le Mhaut, C. Rabut, L.L. Schumaker (Eds.)*, pp. 163–172, 1996.

[12] G. Greiner, L. J., and W. Wesselink, "Surface modeling with data dependent energy functionals," *Computer Graphics Forum*, vol. 15, no. 3, pp. 175–186, 1996.

[13] J. Hoschek and U. Dietz, "Smooth b-spline surface approximation to scattered data," *Reverse Engineering*, pp. 143–152, 1996.

[14] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," *Proceedings of ACM SIGGRAPH*, pp. 173–182.

[15] V. Krishnamurthy and M. Levoy, "Fitting smooth surfaces to dense polygon meshes," *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 313–324, August 1996.

[16] I. Park, I. Yun, and S. Lee, "Constructing nurbs surface model from scattered and unorganized range data," *Proceedings of 2nd International Conference on 3-D Digital Imaging and Modeling . Ottawa, Canada,*, October, 1999.

[17] B. F. Gregorski, B. Hamann, and K. I. Joy, "Reconstruction of b-spline surfaces from scattered data points," *Computer Graphics International*, vol. 18, pp. 163–172, 2000.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Proceedings of ACM SIGGRAPH*, pp. 71–78, 1992.

[19] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, August, 2001.

[20] N. Kojekine, V. Savchenko, and I. Hagiwara, "Surface reconstruction based on compactly supported radial basis functions," *Geometric modeling: techniques, applications, systems and tools*, 2004.

[21] D. R. Forsey and R. H. Bartels, "Hierarchical b-spline refinement," *Proceedings of ACM SIGGRAPH*, vol. 22, pp. 163–172, June, 1988.

[22] D. R. Forsey and R. H. Bartels, "Surface fitting with hierarchical splines," *ACM Transactions on Graphics*, vol. 14, no. 2, pp. 134–161, April, 1995.

[23] R. Pfeifle and H.-P. Seidel, "Fitting triangular b-splines to functional scattered data," *Computer Graphics Forum*, vol. 15, pp. 15–23, June, 1996.

[24] A. Baussard, E. L. Miller, and D. Prmel, "Adaptive b-spline scheme for solving an inverse scattering problem," *Inverse Problems*, vol. 20, pp. 347–365, 2004.

[25] R. Whitaker, "A level-set approach to 3d reconstruction from range data," *International Journal of Computer Vision*, vol. 29, no. 3, October, 1998.

[26] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *Proceedings of ACM SIGGRAPH*, 1996.

[27] J. C. Carr, R. K. Beatson, J. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," *Proceedings of ACM SIGGRAPH*, August, 2001.

[28] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," *Proceedings of ACM SIGGRAPH*, 2000.

[29] J. Davis, S. R. Marschner, M. Garr, and M. Levoy, "Filling holes in complex surfaces using volumetric diffusion," *First International Symposium on 3D Data Processing, Visualization, and Transmission Padua, Italy*, 19-21 June, 2002.

[30] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro, "Impainting surface holes," *International Conference on Image Processing*, August, 2003.

[31] J. Wang and M. M. Oliveira, "A hole filling strategy for reconstruction of smooth surfaces in range images," *XVI Brazilian Symposium on Computer Graphics and Image Processing*, oct 2003.

[32] L. V. Boris Mederos and L. H. de Figueiredo, "Robust smoothing of noisy point clouds," *SIAM Conference on Geometric Design and Computing, Seattle 2003, Nashboro Press.*

[33] C. Lange and K. Polthier, "Anisotropic fairing of point sets," *Special Issue of CAGD*, 2004.

[34] Taubin, "A signal processing approach to fair surface design," *Proceedings of ACM SIGGRAPH*, 1995.

[35] U. Clarenz, U. D. I, and M. Rumpf, "Anisotropic geometric diffusion in surface processing," *EEE Visualization archive, Proceedings of the conference on Visualization*, 2000.

[36] M. Meyer, M. Desbrun, P. Schrder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," *In Visualization and Mathematics III, H.-C. Hege and K. Polthier eds.*, p. 3557, 2003.

[37] H. Zhang and E. L. Fiume, "Mesh smoothing with shape or feature preservation," *In Advances in Modeling, Animation, and Rendering, J. Vince and R. Earnshaw, editors*, pp. 167–182, 2002.

[38] C. L. Bajaj and G. Xu, "Anisotropic diffusion of surfaces and functions on surfaces," *ACM Transactions on Graphics*, vol. 22, pp. 4–32, 2003.

[39] A. Belyaev and Y. O. I.-K. Tel-Aviv, "A comparison of mesh smoothing methods," *Bi-National Conference on Geometric Modeling and Computer Graphics*, pp. 83–87, February, 2003.

[40] J. Peng and D. Zorin, "A simple algorithm for surface denoising," *Proceedings of IEEE Visualization*, 2001.

[41] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transaction on Information Theory*, vol. 41, pp. 613–627, 1995.

[42] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," *ACM Transactions on Graphics*, pp. 943–949, 2003.

[43] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *Proceedings of ACM SIGGRAPH*, p. 950953, 2003.

[44] M. P. Epstein, "On the influence of parametrization in parametric interpolation," *SIAM Journal on Numerical Analysis*, vol. 13, no. 2, pp. 261–268.

[45] M. W. Y and K. J. P, "Parametrization of randomly measured points for least squares fitting of b-spline curves and surfaces," *Computer Aided Design*, vol. 27, no. 9, pp. 663–675, September, 1995.

[46] M. S. Floater and K. Hormann, "Surface parameterization: a tutorial and survey," *Advances on Multiresolution in Geometric Modelling, N. Dodgson, M. S. Floater, and M. Sabin (eds.)*.

[47] M. S. Floater, "Parametrization and smooth approximation of surface triangulations," *Comp. Aided Geom. Design*, 1997.

[48] M. S. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, 2003.

[49] K.Hormann and G.Greiner, "Mips: an efficient global parametrization method," *Curve and Surface Design*, 2000.

[50] B. Lvy, S. Petitjean, R. Nicolas, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *Proceedings of ACM SIGGRAPH*, July, 2002.

[51] A. Sheffer and E. de Sturler, "Parameterization of faceted surfaces for meshing using angle based flattening," *Engineering with Computers*, vol. 17, pp. 326–337, 2001.

[52] M. S. Floater and M. Reimers, "Meshless parameterization and surface reconstruction," *Computer Aided Geometric Design*, vol. 18, pp. 77–92, 2001.

[53] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe, "Signal-specialized parametrization," *Eurographics Workshop on Rendering*, 2002.

[54] L. S. Tekumalla and E. Cohen, "Smoothing space curves using the moving least squares projection," *Geometric Modeling, Visualization and Graphics*, 2005.

[55] L. S. Tekumalla and E. Cohen, "A hole-filling algorithm for triangular meshes," Tech. Rep. UUCS-04-019, University of Utah, 2004.