Comenius University, Bratislava, Slovakia
Faculty of Mathematics, Physics and Informatics
Department of Geometry

Marián Uherčík:

Implicit Surface Reconstruction using Local Approximations from Unorganized Set of Points

(Master Thesis)

Advisor:
RNDr. Pavel Chalmovianský, PhD.

Bratislava,
April 2005

I honestly declare that I have written the submitted master thesis by myself and I have used only the literature mentioned in the bibliography.

Bratislava, 29th April 2005 .....

Marián Uherčík

I would like to thank to my master thesis advisor Pavel Chalmovianský for his valuable advices, remarks and suggestions.

I would like to thank also to my family for supporting me and for their patience during my work.

Abstract

This work considers the problem of surface reconstruction from unorganized set of points. We use local polynomial approximations and we blend them to the implicit surface. We survey the recent work in this area. We discuss the sampling requirements, various space subdivisions, local approximations classes and blending techniques. We implement an octree based algorithm based on work of Ohtake et al. that allows us to contruct error controlled surface reconstruction. We discuss the problem of sharp features reconstruction. We extend the model of implicit surface by minimum and maximum functions capable to model sharp local features. Finally we give a technique for fairing of previously reconstructed local features to eliminate the errors of certain type.

Keywords: Implicit Surface Reconstruction, Local Approximations, Sharp Features Reconstruction, Multi-level Partition of Unity Implicits, Edges and Corners

Abstrakt v slovenčine. Táto práca rieši problém rekontrukcie plochy z množiny bodov. Používame lokálne polynomiálne aproximácie, ktoré stmelujeme do implicitnej plochy. Práca obsahuje prehľad literatúry. Rozoberáme požiadavky na vzorkovanie, rôzne techniky na prerozdelenie priestoru, rôzne typy lokálnych aproximácií a stmeľovanie. Implementujeme algoritmus (pochádza od Ohtake-ho) založený na oktálovom strome, ktorý dovoľuje adaptívnu chybou riadenú rekonštrukciu plochy. Rozoberáme tiež problém rekonštrukcie ostrých čŕt. Rozširujeme model implicitnej funkcie o minimum a maximum tak, že sme schopný rekonštruovať ostré lokálne črty. Nakoniec uvádzame techniku pre úpravu rekonštruovaných lokálnych čŕt za účelom odstránenia chýb určitého typu.

Kľúčove slová: Rekonštrukcia implicitnej plochy, Lokálne aproximácie, Rekonštrukcia ostrých čŕt, Multi-level Partition of Unity Implicits, Hrany a rohy

Contents

List of Figures

Chapter 1

Introduction

The creation of 3D content is the goal of geometric modeling and it is deeply studied in many ways. To solve this problem algorithmically, the various graphics pipelines has been designed. Depending on the primitives used, it is from 0-dimensional (0D) points, 1D wire-frame, 2D boundary representation including mesh and parametric surfaces, and at last 3D volumetric approach which is represented by CSG or implicit surfaces. To create a 3D model we can use standard geometric modeling techniques which are widely studied domain.

The alternative to this artificial process is to reconstruct it from measured data which is a natural way how to get 3D model from physical world. This process is converse to the direction of modeling pipeline. We want to reconstruct the original object. To do it we infer the parameters which are the best description of original object. When the pipeline is restarted with reconstructed parameters we should get the result similar to the original.

We view the surface reconstruction in the context of modeling of physical forms such as image, movie, sound, shape. In the life cycle of modeling of such form, there are two important transformations: *acquisition* to an intermediate form (representation) and *instantiation* back to the physical form. The degree of successfulness always depends on the amount of user interaction. In the 3D shape acquisition, the fully automatic process needs the collaboration of the 3D scanner device and the surface reconstruction method. The instantiation transformation of 3D shape can be executed by NC milling or stereo lithography.

Why Implicit Surfaces? We decide to choose implicit surface as the main surface representation because in contrast of parametric surfaces it has many good properties. For example it has easy accessing to evaluation, good modeling properties, good extrapolation features and the ability to represent object of any topology. The drawback of implicit surface is a problematic rendering. Best frequent methods are conversion to polygonal model, but it must be restarted each time when the original implicit surface is changed.

From this follows unpractical user interaction and using it in virtual reality. The discussion about the alternative representations of implicit surfaces are in section 3.1.4.

1.1 Problem Statement

As input, we have an unorganized set M of points in 3 dimensional euclidean space E^3. The input set can include some error. The points come from a sampling process, for example merged range data from 3D scanner:

$$M = \{ \mathbf{x_i} \mid \mathbf{x_i} \in \mathcal{R}^3,\ i = 1, \ldots, n \}.$$

We want to reconstruct surface and to have a mathematical model that approximates the points. The approximation will not be necessarily the best one but we want to find an algorithm that is reasonable for use. The more exact conditions are stated bellow. We consider that points are well sampled from surface of a single object. It means that points are sufficiently dense sampled and with low noise. We use implicit surfaces. Hence, the output is a function

$$f : \mathcal{R}^3 \rightarrow \mathcal{R}$$

that represents a volume object. The surface is defined as the zero-level set

$$S = \{ \mathbf{x} \in \mathcal{R}^3 \mid f(\mathbf{x}) = 0 \}$$

and should be an approximation to the set of points M.

1.1.1 Main Requirements

We use the local approximations because this way we get a true picture of the shape even we are using an approximation technique.

Topology. We want to be able to reconstruct an object of any topology. The implicit surfaces have this ability.

Sampling. We want to manage the variability of the sampling and we want to cover the small holes.

Numerical Stability and Noise resistance. We want to have an algorithm that is numerically stable. It means that small change in input data does not affect reconstructed surface.

To manage it we need preserve the noise at reasonable level. With this knowledge we can control and adjust the reconstruction algorithm to get better results with respect to the noise.

1.1.2 Fundamental Steps

We give a brief outline of algorithm. It works in two phases:

1. Approximate surface locally, most likely with an algebraic surface of the low degree.
2. Blend them to one global smooth implicit function. This step is evaluated each time when the global implicit function is evaluated.

1.1.3 Related Subproblems

To design a solution we deal with several principal subproblems.

Space subdivision technique. We use a technique to divide space to smaller parts to better reconstruction of small details and to get fast algorithm. We want to use the multi-resolution technique. The octree technique is outlined as a good choice but it is not geometric invariant to the rotation.

Local approximation. We use an algebraic surface of the low degree. The normalization is needed to avoid degeneration of the solutions. Then, we use standard numerical method for solving the least-squares problem. The fitting can produce artifacts such as singularities or additional unwanted branches. We need to avoid them.

Blending. We need the blending to obtain the smooth result. This method blends the neighboring local approximations to smooth surface preserving the continuity of any order – it depends only on the blending functions that were used. Furthermore, if the compactly supported basis of the blending functions is used then it preserves the principle of locality.

Edges and corners. To improve the quality of approximation we may use non-polynomial approximation functions. For the objects with sharp edges and corners we extend the implicit surface model. The features reconstruction is one difficult problem in reverse engineering.

Normals estimation. Normals are important in reconstruction not only for resolving orientation of object. They help to obtain more accurate result and help to detect sharp features. The 3D scanning often produces normals, but if they are not included in the input data, then we need to estimate them.

1.2 Applications

There are numerous applications of implicit surface reconstruction techniques.

3D scanning. The reconstruction of surface from range data that come from the optical triangulation [Cur97], is an easy way how to obtain the 3D model of any object. Nowadays, the scanning devices produce accurate high-density samples so this way is an alternative to computer modeling of virtual object.

Computer Aided Geometry Design. The modeling by sample points is an alternative way to CAGD parametric surfaces. Since it abstracts from the many parameters this approach is more robust and user friendly than standard CAGD.

Point based CG. The rendering point clouds, re-sampling point set surfaces, hole filling (tutorial on Eurographics 2002 [AGP+02] is about Point-based Computer Graphics), in Pauly's dissertation thesis [Pau03] is studied point primitives in context of computer graphics pipeline.

Volume description. In the medicine area, it is used for volumetric visualization of data acquired from various measurement devices (Computer Tomography, Ultrasound, X-rays). Using an implicit function, we can describe also natural phenomena in meteorology and run computer simulations.

The following examples are specific applications.

Digital Michelangelo. It is an interesting project [Mic] at Stanford University computer graphics lab. The goal was to reconstruct the statue of David. It was a great technological challenge. The large amount of range data was generated so the accuracy of sampling was at a very good level.

3D fax project. It is an idea how to transfer physical form from one place to another through data channel for example a computer network. It was already mentioned in the previous section, the description can be found in [Cur97].

1.3 Thesis Outline

This thesis is organized as follows:

- Chapter 1 introduces to the problem of surface reconstruction. Chapter 2 gives a brief survey of selected important previous work in surface reconstruction depending on the various representations: simplicial, parametric and implicit.
- Chapter 3 considers the implicit surface reconstruction problem and discusses the related subproblems, it gives a possible solution for each subproblem.
- Chapter 4 considers the feature detection and reconstruction problem with implicit surfaces, especially the sharp features like edges and corners. We propose an original solution for this problem.
- Chapter 5 concludes the thesis by giving results, summarizing the contribution and outlining the future work.
- Appendix A outlines the software specification.
- Appendix B contains a description of the included compact disc.

Chapter 2

Recent Work

There are several groups of interest in surface reconstruction depending on the representation used – simplicial, parametric, implicit surfaces. Even though there are various representations, there are some common problems. We are going to give a brief survey to the surface reconstruction techniques, with focus on the implicit ones.

We discuss how to get local approximations of whole surface. We concentrate on two ways how to get smooth implicit surface. The first case is to connect small patches that must satisfy continuity conditions. The construction of such constraints can be problematic. The second case is to blend patches assuming the patches overlap.

2.1 Simplicial Surfaces

The reconstruction of simplicial surfaces can be defined as finding the subgraph of Delaunay complex of points [EM94]. Many algorithms follow this idea and differ mostly in how they identify triangles that belong to surface.

Crust algorithms [ABK98] use vertexes of Voronoi diagram as an approximation to medial axis. Sufficient sampling guarantees a reconstruction of the original topology. Cocone algorithm [DGH01] has similar guarantees but eliminates the step of adding Voronoi vertexes to the point set. Still, constructing the Delaunay complex of millions of points is costly and some algorithms rather use local triangulations of the points [BMR+99].

Hoppe et al. [Hop94] defines an implicit function that is dual to Delaunay-type reconstruction: For each point a normal direction is estimated from neighboring points and all normals are oriented consistently. The signed distance to surface is defined as normal component of the distance to the closest point. Thus, the surface consists of planes through the points bounded by the Voronoi cells of the adjacent points.

In many practical cases, one has multiple point samples for the same region. A way to consolidate this information is to build a distance function in volumetric grid by properly

weighting the points [Cur97].

2.2 Parametric Surfaces

The simplicial surface is defined as a set of planar pieces resulting in C^0-continuous approximations. To achieve smoother approximations, one can either build a smooth surface over the triangulation [Hop94] or fit smoother functions but when there is a vertex of degree 4 and higher, then raises the problem of guaranteeing the C^e-continuity, for $e \geq 1$.

There are methods that reconstruct the smooth parametric surfaces. The object domain has to be divided into smaller parts because the parametric surface cannot represent the object of more complex topology. To ensure the continuity there must be set some border conditions. The problem defined in such way has been examined but however there exist some solutions they are not direct [Hop94].

2.3 Implicit Surfaces

Most implicit shape reconstructions from point sets are based on Blinn's idea of blending local implicit primitives [Bli82]. Blobs, as he called the implicit surfaces, create an interesting meta-balls blending effect and were used to model molecular shapes.

The implicit surfaces are now widely used in surface reconstruction methods. They have several benefits like very good modeling properties or ability to represent surface of any topology.

In [JF02], the points and associated normals are simultaneously approximated. The described algorithm is geometrically invariant, it requires no artificial normalization in parameter space and the normalization is done by normals. There are also some error bounds for approximation.

There is much work in global algebraic surface fitting that uses one polynomial to fit the point cloud. Taubin in [Tau91] has reconstructed algebraic curves and surfaces in 2D and 3D space. He has defined an improved algebraic distance function normalized by gradient. This is an approximation to the Euclidean distance. This method is invariant under rigid transformations. He has proved that problem of implicit polynomial fitting leads to generalized eigenvalue problem.

More effective reconstruction algorithm can be achieved by space domain decomposition. Bajaj et. al uses piecewise algebraic curve segments defined on triangles in Bernstein-Bézier basis. The continuity and smoothness is gained by blending on near borders or by constraints to derivative on borders. Self-contained material is given in the trilogy [XBX00, XBC00, BX01].

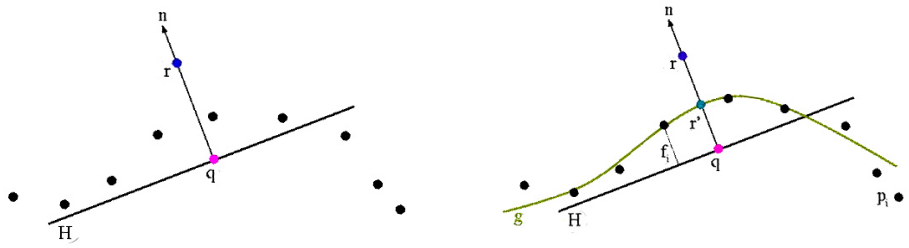


Figure 2.1: 2D Illustration of MLS Projection

Savchenko et. al. [SPOK95] analyze using F-representation in surface reconstruction from scattered surface points and contours. The discussion about F-representation is in section 3.1.4.

In next sections, we describe the three most used methods in implicit surface reconstruction – Moving Least Squares, Radial Basis Functions and Partition of Unity Implicits. All of them uses the paradigm of space decomposition and local approximations.

2.3.1 Moving Least Squares

Moving least squares (MLS) method fits globally smooth function locally and then blend these local surface approximations. MLS approximation brings it to the extreme by building a local fit for every point on the surface. We give the definition from [Pau03] in chapter 2, section 3.

Given a set $P = \{\mathbf{p_i}\}$, the MLS surface $S(P)$ is defined as the stationary point set of projection operator ψ_P, i.e.,

$$S(P) = \{\, \mathbf{x} \in \mathcal{R}^3 \mid \psi_P(\mathbf{x}) = \mathbf{x} \,\},$$

where $\psi_P(\mathbf{x})$ is defined by a two step-procedure illustrated on the figure 2.1:

1. Compute local reference plane (defined by normal $\vec{n} \in \mathcal{R}^3$ and offset $d \in \mathcal{R}$)

   $$H = \{\, \mathbf{y} \in \mathcal{R}^3 \mid \mathbf{y} \cdot \vec{n} - d = 0 \,\}$$

   by minimizing the weighted sum of squared distances

   $$\sum_{i \in I} (\, \mathbf{p_i} \cdot \vec{n} - d \,)^2 \, \phi(\, ||\mathbf{p_i} - \mathbf{q}|| \,),$$

   where $\mathbf{q}$ is orthogonal projection of $\mathbf{x}$ onto H. The reference plane defines a local coordinate system with $\mathbf{q}$ at origin. Let (u_i, v_i, f_i) be the coordinates of the point $\mathbf{p_i}$ in the coordinate system, i.e. (u_i, v_i) are the parameter values in H and f_i is the height of $\mathbf{p_i}$ over H.

2. Compute a bivariate polynomial $p(u, v)$ that minimizes

$$\sum_{i \in I} (p(u_i, v_i) - f_i)^2 \, \phi(||\mathbf{p_i} - \mathbf{q}||).$$

The kernel function ϕ controls the shape of the surface $S(P)$. A suitable choice for ψ is the Gaussian, i.e., $\phi(x) = e^{-\frac{x^2}{h^2}}$, $x \in \mathcal{R}^3$ where h is a global scale factor that determines the kernel width.

This method can be used for noise removal and object smoothing. It is used in point set surfaces [AGP+02] to object re-sampling and rendering. It allows also accurate raytracing of point-based objects. Foundations of this method can be found in the [ABCO+02].

2.3.2 Radial Basis Functions

Radial Basis Functions (RBF) rise by assigning local approximations to points in space but they often use non-compactly defined blending functions. This computation traditionally involves the solution of large linear system, however, it is nowadays tackled using compactly supported functions, multiple expansions, thinning, or hierarchical clustering.

We explain this method according to [CBC+01]. Given a set of distinct nodes $P = \{\mathbf{p_i}\}_{i=1}^N \subset \mathcal{R}^3$ and a set of function values $F = \{f_i\}_{i=1}^N \subset \mathcal{R}$, we need to find an interpolant $s : \mathcal{R}^3 \to \mathcal{R}$ such that:

$$s(\mathbf{p_i}) = f_i, \; i = 1, \ldots, N$$

The set $P = \{\mathbf{p_i}\}$ contains zero-valued surface points $f(\mathbf{p_i}) = 0$, $(i = 1, \ldots, n)$, and the off-surface points $f(\mathbf{p_i}) = d_i \neq 0$, $(i = n+1, \ldots, m)$. We construct a RBF function of this form:

$$s(\mathbf{x}) = p(\mathbf{x}) + \sum_{i=1}^{N} \lambda_i \, \phi(|\mathbf{x} - \mathbf{p_i}|),$$

where p is a polynomial of low degree and the basic function ϕ is a real-valued function on $[0, \infty)$, usually unbounded and with non-compact support. In this context, the points $\mathbf{x_i}$ are referred to as the centers of the RBF.

Popular choice for basic function ϕ is the Gaussian $\phi(r) = e^{-cr^2}$, where c is a constant. For purpose of extrapolation of incomplete input data, it is suitable to choose non-compactly supported basic function, for example $\phi(r) = r$.

We find the solution by minimization of thin-plate energy function which measures the deviation from linear surface and therefore it expresses the smoothness of the function $s(\mathbf{x})$:

$$\int_{\mathbf{p} \in \mathcal{R}^3} \left(\frac{\partial^2 s(\mathbf{p})}{\partial x^2} \right)^2 + \left(\frac{\partial^2 s(\mathbf{p})}{\partial y^2} \right)^2 + \left(\frac{\partial^2 s(\mathbf{p})}{\partial z^2} \right)^2 + 2 \left(\frac{\partial^2 s(\mathbf{p})}{\partial x \partial y} \right)^2 + 2 \left(\frac{\partial^2 s(\mathbf{p})}{\partial x \partial z} \right)^2 + 2 \left(\frac{\partial^2 s(\mathbf{p})}{\partial y \partial z} \right)^2 dv.$$

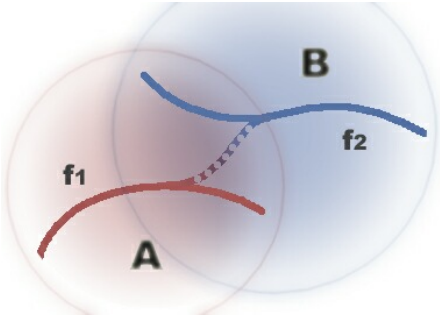


Figure 2.2: An Example of Radial Basis Functions

For polynomial p of degree m, this leads to boundary conditions:

$$\sum_{i=1}^{N} \lambda_i \, q_(\mathbf{p_i}) = 0, \text{ for all polynomials } q \text{ of degree at most } m$$

These boundary conditions along with interpolation conditions lead to linear system for the coefficients that specify the RBF.

Because of global fit the computation costs is high, and nowadays, it is still unusable to bigger data acquired from high-density range scans. There are many methods how to speed up the evaluation of RBF, we cite the Fast Multipole Method [BG97]. There are also some new methods how to construct adaptive approximation with RBF [OBS04]. Turk examines the modeling properties of RBF based on interpolation [TO02].

2.3.3 Multi-level Partition of Unity

The work [OBA+03] combines more techniques previously mentioned techniques. The detailed description of this technique is in chapter 3.6 so here we give only three key ingredients to this approach:

- Piecewise quadratic functions that fit the local shape of the surface.
- Weighting functions (partitions of unity) that blend them together.
- An octree subdivision method that adapts to variations in the complexity of the local shape.

This approach gives the flexibility in the choice of local shape functions. Moreover, it is able to represent sharp features such as edges and corners by an extension of shape functions.

Ohtake et al. continue on his own work and construct adaptive RBF surface [OBS04] that uses MPUI for coarse approximation and Xie et al. [XMQ] extends this work.

2.4 Features Reconstruction

We mention some papers related to reconstruction of features. Not all papers are dealing with our problem of feature reconstruction in the context of implicit surfaces.

Already mentioned MPUI implicits can reconstruct sharp local features such as edges and corners. It uses the work of Kobbelt et al. [KBSS] and he solves the problem of feature preserving in polygonal model by extending implicit function by normals and then the extended marching cubes algorithm can polygonize the sharp features more accurately.

Pauly et al. [PKG03] is inspired by the rule: *"If the edge occurs in different scales, then the edge corresponds to the real edge"*. They do it by local analysis the surface by Principal Component Analysis and they reconstruct the feature lines. They have also implemented a non-photorealistic point renderer to visualize point set surfaces.

Song et al. [SVD02] proposes a free-form surface design based on feature reconstruction. He introduces a free-form feature library that is a basis of the system. The features can either be extracted from existing regions or be constructed by designers. The effective free-form designs are constructed by boolean-like operations using the library of features.

Varadhan et at. [VKKM] designs a isosurface reconstruction sensitive to features. He achieves it by using a contouring algorithm on the Kobbelt's directed distance field. For boolean operations he uses an adaptive subdivision sensitive to sharp features.

Chapter 3

Implicit Surface Reconstruction

3.1 Introduction to Implicit Surfaces

3.1.1 Basic Definitions

The implicit surfaces in geometric modeling raised from the idea of Blinn's blobby models [Bli82]. The fundamentals about implicit surfaces is can be found in Bloomenthal's book [Blo97]. We follow the definitions quoted here.

We define the implicit surface implicitly as a zero-level set of an implicit function, in contrast to the parametric surface that is embedding of surface to 3D space. First, we define the implicit surface function, also known as scalar field or field function.

Definition 1 *Implicit Surface Function is function* $F : \mathcal{R}^3 \rightarrow \mathcal{R}$. *The function* F *must be at least* C^0 *continuous.*

We can evaluate it at any point $\mathbf{p} = (x, y, z) \in \mathcal{R}^3$ to return a scalar value $F(x, y, z)$. The function F is usually C^2, that means the first or the second order derivatives exist for each point $\mathbf{p}$.

Definition 2 *Implicit Surface* S *is a set of all points* $\mathbf{p} \in \mathcal{R}^3$ *that evaluates to zero, i.e.* $F(\mathbf{p}) = 0$.

Hence, the notation $S = F^{-1}(0)$ can be used instead of S. Points in space $\mathbf{p} \in \mathcal{R}^3$ not on the surface S can be divided into two categories according to the positive or the negative evaluation into: the inside and the outside of object.

Definition 3 *The set of points* $\mathbf{p} = (x, y, z)$, *such that* $F(x, y, z) > 0$ *is called interior volume of* S.

F is typically specified either by mathematical functions, in which one or more equations is evaluated or by procedural methods, in which an algorithmic process evaluates F. Our primary representation use functional representation composing polynomials and building the whole surface. The evaluation of functional representation is done using an algorithmical process.

The function F can be extended by other attributes defining additional implicit function, like color $F_{color} : \mathcal{R}^3 \rightarrow RGB$, or gradient $F_{gradient} : \mathcal{R}^3 \rightarrow \mathcal{R}^3$, and so on.

Example. We introduce an example of comparison of parametric representation to implicit representation. For example let us compare a hemisphere definition with unit radius in parametric form

$$p_x(u,v) = \frac{(1-u^2)(1-v^2)}{(1+u^2)(1+v^2)},\ p_y(u,v) = \frac{2u}{(1+u^2)},\ p_z(u,v) = \frac{2v}{(1+v^2)},\quad u,v \in [-1,1]$$

and a sphere with unit radius in implicit form which is simpler

$$f(x,y,z) = x^2 + y^2 + z^2 - 1 = 0,\quad (x,y,z) \in \mathcal{R}^3$$

As this example shows, the implicit representation of the surface is often simpler than the parametric form.

3.1.2 Continuity and Differentiability

For a given point in space $\mathbf{p}$, the partial derivatives define the gradient $\nabla F = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z})$ of the function F. If the gradient is non-zero at a point $\mathbf{p}$, then $\mathbf{p}$ is said to be *regulars*. The unit-length gradient $\nabla f(\mathbf{p})$ in a regular point p is *normal* to the surface at p. If the gradient is equal zero, then the point is *singular*. If the whole surface is regular, it is a *manifold*.

Implicit Function Theorem. *Unbounded 2-manifold* is a surface embedded in $\mathcal{R}^3$ such that a neighborhood around any interior point on the surface is homeomorphic to a disk. The manifold can be *bounded* (or *closed*), then the boundary parts can be homeomorphic to an half-disc. All other surfaces are *non-2-manifold*.

The next theorem says that when we take first derivatives of nonsingular implicit surface, then we can use a parametric surface to describe the surface for small neighborhood of point $\mathbf{0}$. We note that arbitrary point can be chosen in which the implicit function is regular.

Theorem 1 *For every function $f : \mathcal{R}^n \rightarrow \mathcal{R}$ which is C^k-continuous ($k \geq 1$) such that satisfies $\nabla f(\mathbf{0}) \neq \mathbf{0}$ and $f(\mathbf{0}) = 0$, exists such small $\varepsilon > 0$ and a unique function $p : (-\varepsilon, +\varepsilon)^{n-1} \rightarrow \mathcal{R}^n$, such that p is C^k-continuous, and $f(p(t)) = 0$.*

Figure 3.1: Modeling with Implicit Surfaces

3.1.3 Modeling with Implicit Surfaces

From the implicit surface function can be determined not only the surface but also the solid object and algebraic distance. From this fact follow some useful modeling properties of implicit surfaces.

We divide the modeling functions in dependence on the count of input arguments into *nilary*, *unary*, *binary*, and the others. We consider each implicit surface as *nilary* if it does not need any input argument and we view it as original surface. We call *unary* modeling functions as deformators because it usually takes one implicit surface and transforms it into another shape. It is for example adjusting isolevel, smoothing, scaling and skewing.

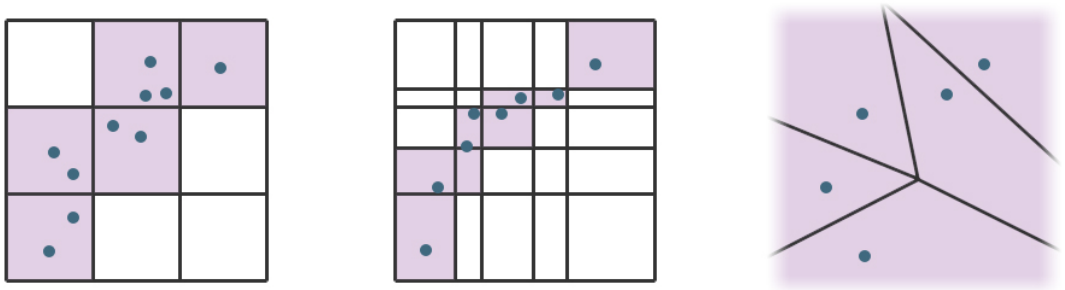
With the *binary* modeling functions can be easily modeled boolean operations such as union, difference, intersection. They are capable to blend two surfaces either in space or in time. Using *min* and *max* functions we can model the sharp edges that are to be used in feature modeling.

Adding more dimensions to input arguments of implicit surface we are able to model surface material and also the change of object in time and create animations.

On the figure 3.1, the examples of various modeling operations like difference, skewing and union.

3.1.4 Alternative Representations

Because of problematic rendering of the implicit surfaces there is always appearing a question if there is more suitable representation. Leaving the model in implicit representation does not allow a real-time interaction. We can get high-quality output by raytracing but if we want to change the view we need to start it again and it is very slow because it evaluates the implicit surface for each pixel to find the solution of algebraic equation implied by the implicit surface.



uniform grid non-uniform grid Voronoi diagram

Figure 3.2: Various Spatial Subdivisions in 2D

Polygonal

For many applications, it is useful to approximate an implicit surface with a mesh of triangles or polygons. Well known polygonization algorithms are *marching cubes* [Blo87]. They use exhausting evaluation of space so its complexity is in worst case cubic in number of samples in one dimension. The result can be used for user interaction like view transform or further modeling. *Marching triangles* [HI97, Har98, SA01] algorithm generates more uniform set of triangles. It needs to start from initial seed point and its complexity is $O(\frac{S}{s})$, where S is the size of surface and s is size of basic triangle.

F-rep

More advanced representation than black-box approach is *Functional representation*. This idea has been developed by A. Pasko et al. [PASS95]. They have developed the language *Hyperfun* to describe the implicit surface for geometrical modeling. To visualize the model, they use Bloomenthal's polygonizer [Blo87] or POV-Ray [POV] ray tracer. Hasan [Has03] solves the problem of visualization of certain subclasses of implicit surfaces. He achieve near-realtime visualization by precompiling Hyperfun model.

3.2 Space Subdivision

We assign a domain of effect for each local approximation. Hence we build decomposition of space $U \subset \mathcal{R}^3$ to *elementary cells* $\{U_i\}_{i=1}^{W}$. The decomposition must cover the whole space domain $\bigcup_{i=1}^{W} U_i \supset U$. It can be ambiguous, so for one point $\mathbf{p}$ in the space U there can be more than one local approximation $U_{i_1}, \ldots U_{i_k}$. The overlapping of local approximation is often desired effect because further blending produces more reliable result.

3.2.1 Uniform Subdivision

It works without any additional information about sampling of points. Each elementary cell is of the same shape or there are more than one elementary cells that regularly vary over space. We can change the density of cells by adjusting the size and shape of elementary cell. Some cells may remain empty. We omit computing approximations for such cells. We estimate the computation complexity for computing the approximations for all non-empty cells. It is $O(n^2)$, where n is number of samples in one dimension because we consider the surface to be 2-manifold.

Regular Grid. Regular grid decomposes space U to uniformly positioned cubes or boxes of equal size. If we consider the space to be a unit cube $U = [0, 1]^3$, then the decomposition is

$$U = \bigcup_{i=1\ldots p,\ j=1\ldots q,\ k=1\ldots r} \left[\frac{i-1}{p}, \frac{i}{p}\right) \times \left[\frac{j-1}{q}, \frac{j}{q}\right) \times \left[\frac{k-1}{r}, \frac{k}{r}\right).$$

We adjust the size of box element by adjusting the number of cells $p, q, r \geq 1$ in one dimension. The dimensions of each box is $(\frac{1}{p}, \frac{1}{q}, \frac{1}{r})$. The task is to adjust the level of density of grid to achieve the best effect. We can estimate the density of grid from count of all points.

3.2.2 Non-uniform Subdivision

In this type of space subdivision, we consider also points and their normals. This type of subdivision is able to adapt to varying density of sample points. Similarly to uniform subdivision we omit the empty cells.

Non-uniform Grid. Uses non-equal sized boxes for elementary cells. It is defined using grid lines $s_x = \{x_0, \ldots, x_p\}$, $s_y = \{y_0, \ldots, y_q\}$, $s_z = \{z_0, \ldots, z_r\}$, where $x_0 < x_1 < \ldots < x_p$, $y_0 < y_1 < \ldots < y_q$, $z_0 < z_1 < \ldots < z_r$. The decomposition of domain U is

$$U = \bigcup_{i=1\ldots p,\ j=1\ldots q,\ k=1\ldots r} [x_{i-1}, x_i) \times [y_{i-1}, y_i) \times [z_{i-1}, z_i).$$

The estimation of $p, q, r \geq 1$ and the placement of grid lines should respect the distribution of points in each dimension.

3.2.3 Hierarchical Subdivision

Above mentioned subdivision techniques offer no feedback. With the hierarchical subdivision we can refine the global approximation on selected place without affecting the rest of local approximations. This technique allows us to construct iterative approximation that adapts to the error of previous approximation.

Octree. This is a well-known tree structure. Each node of shape of cube $N = [x_0, x_1] \times [y_0, y_1] \times [z_0, z_1]$ has either 8 children $N_1, \ldots, N_8$ with equal dimensions or it is a leaf with no children. Domain $U = [0, 1]^3$ is the single root.

This approach is simple and therefore it allows us to construct efficient algorithm.

3.2.4 Voronoi Diagram

Voronoi diagram is a theoretical structure well-known from computational geometry [CFGN01]. For the set of distinct points $\{\mathbf{p_i}\}_{i=1}^{n}$ in space $U \subset E^d$, it decompose U to Voronoi cells $\bigcup_{i=1\ldots n} V(\mathbf{p_i})$, where $V(\mathbf{p_i}) = \{\mathbf{r} \in U | \forall j = 1 \ldots n, j \neq i : d(\mathbf{r}, \mathbf{p_i}) \leq d(\mathbf{r}, \mathbf{p_j})\}$ is a part of space with the nearest point $\mathbf{p_i}$.

Vector quantization [VQ] was originally made for lossy data compression. It computes Voronoi diagram for the case with number of cells m less than number of points n. The resulting decomposition contains approximately equal number of points in each Voronoi cell.

Hence, the Voronoi diagram is an ideal structure for space decomposition. But the computation costs are too high for using it for the surface reconstruction because it works with many input points. The heuristic space decomposition (such as Radial Basis Functions, section 2.3.2) that respects the rule of approximatively uniform space subdivision intuitively approximates the Voronoi diagram structure.

3.3 Sampling Requirements

A major difficulty is that the topological type of surface is unknown and must be inferred from the points. Nowadays 3D scanners generate large amount of sample points, so the task of inferring topological type can be managed. The goal is to design sufficient conditions that must be satisfied to run algorithm successfully and be able to correctly infer the topological type.

Our task is for assumptions given in section 1.1.1 to quantify the maximum allowed level of noise. The density of sampling can vary over space. We notice that first item is solved by the construction of algorithm and choosing appropriate representation of surface.

We follow the conception presented in [Hop94, p. 19-21]. It answers to the question of minimal sampling density and maximum allowed noise level.

Sampling Density and Noise Level. To assume the sample points to be noisy means that each point $\mathbf{x_i} \in X$ can be moved at most by distance δ. Such sample X is called δ-noisy. To capture the notion of sampling density we use following definition. Let Y be a noiseless sample of unknown surface U. The sample Y is said to be ρ-dense if any sphere with radius ρ and center in U contains at least one point in Y. A δ-noisy sample $X \subset \mathcal{R}^3$

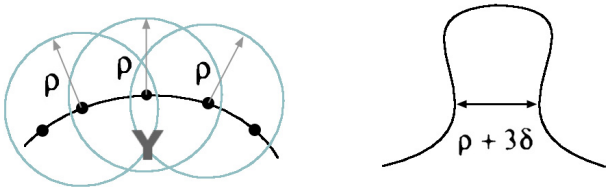


Figure 3.3: Assumption on the Size of Features

of surface U is said to be ρ-dense if there exists a noiseless ρ-dense sample $Y \subset U$ such that points of Y originate from noiseless sample X.

Let us consider the set of sample points X. For noiseless sample we can infer that $\mathbf{p}$ with $d(\mathbf{p}, X) > \rho$ cannot be a point of U. Intuitively, the sample points do not leave holes of radius larger than ρ. If the sample is δ-noisy, the radius of the holes may increase. We therefore conclude for given a ρ-dense, δ-noisy sample X of U, a point p cannot be a point of U if $d(\mathbf{p}, X) > \rho + \delta$.

Size of Features. Small details (compared to either ρ or δ) can be left. The critical is to infer correct topology. We must assume that no two parts of U come too close together. In other words, no two sheets of U may come within distance $\rho + 3\delta$ of each other.

An estimation of δ can be computed in most applications (e.g., the accuracy of the laser scanner). Similarly, analysis of the scanning procedure can also provide an estimation for the sampling density ρ.

3.4 Local Approximations

We give a theoretical introduction and then discuss various approximation techniques. This is used for local approximation – each local approximation is fitted as an algebraic surface.

Weierstrass Theorem This is fundamental knowledge of approximation theory. It says that there are no theoretical obstacles to construct a good algorithm.

Theorem 2 *(Weierstrass, 1885): Let f be a continuous real function $f : \mathcal{R} \to \mathcal{R}$. Then, for every $\varepsilon > 0$, there is a polynomial p such that $\forall \mathbf{x} \in \mathcal{R} : | f(\mathbf{x}) - p(\mathbf{x}) | < \varepsilon$.*

Every smooth function G can be approximated by infinite set of polynomials g_i, or equivalently function G can be approximated with demanded error by finite set of polynomials g_i. There are also approximations based on non-polynomial basis, for instance trigonometric series and the Fourier series analysis. We leave the discussion about these theorems and

refer reader the relevant literature [Car98]. Our goal is to give algorithm for solving surface reconstruction problem.

3.4.1 Parameter Space

For better understanding of the surface fitting we provide some facts about the parameter space of surfaces that we fit. We give definitions inspired by [Pra87].

To define the plane we need 3 non-collinear points, to define the sphere or the axis-oriented ellipsoid we need 4 non-coplanar points. Generally the implicit polynomial surface can be defined by certain sample of points. When the number of points is more than degree of freedom of surface, then no exact fit is possible. We want to find the surface that is as close to the sample points as it gets.

Definition 4 *Basis of parameter space is a set of functions* $\{g_i : \mathcal{R}^3 \to \mathcal{R} \mid i \in I\}$ *such that* g_i *is independent mutually, i.e. no one is can be expressed as linear combination of others. Set of all linear combinations constitute the parameter space* $\mathcal{G}_I$*.*

Correctly defined parameter space $\mathcal{G}_I$ for finite index set I constitutes a vector space so we identify parameter space with parameters. System $\mathcal{G}$ also constitutes a ring, we call it a *shape*. Subsets $\mathcal{H} \subset \mathcal{G}$ can be defined as an ideal of this ring, we call it a *subshape*.

Parameter space of implicit polynomials of degree at most d can be defined using polynomial basis:

$$Q^d(c_{ijk})(x, y, z) = \sum_{0 \le i+j+k \le d} c_{ijk} x^i y^j z^k$$

Set of all combinations of $c_{i,j,k}$ constitutes its parameter space $\mathcal{Q}^d$.

As an example sphere shape $\mathcal{S}_{sphere}$ can be defined as subshape of general shape $\mathcal{Q}^2$ by basis $\{x^2 + y^2 + z^2, x, y, z, 1\}$, or shape of cylindrical surface aligned to z-axis $\mathcal{R}_{cylinder}$ by basis $\{x^2 + y^2, x, y, 1\}$.

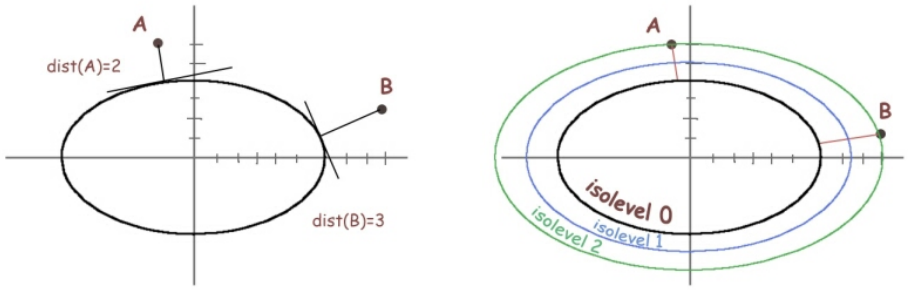
To achieve best reconstruction it helps to have some additional information such as normals. The knowledge of shape is also one of the additional information. Generally, by choosing an appropriate basis we get more accurate result. An additional condition for improvement the stability can be the orthogonality of the polynomials in the basis.

3.4.2 Distance Measurement

To build the metric space we define general distance of two points. Later, the distance of point to the implicit surface is introduced.

Definition 5 *The general distance function of two points* x *and* y *in space* U *must satisfy*

1. $dist(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$ for all $\mathbf{x}, \mathbf{y} \in U$



euclidean distance algebraic distance

Figure 3.4: Euclidean and Algebraic Distance

2. $dist(\mathbf{x}, \mathbf{y}) \geq 0$ for all $\mathbf{x}, \mathbf{y} \in U$

3. $dist(\mathbf{x}, \mathbf{z}) \leq dist(\mathbf{x}, \mathbf{y}) + dist(\mathbf{y}, \mathbf{z})$ for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in U$

For the purposes of surface fitting we need to measure the distance of point to the surface. We give two approaches how to do it: the Euclidean distance and the algebraic distance (figure 3.4).

Euclidean Distance. This is the intuitive notion of this the distance. To find the distance of point p to surface S, we find the point $\mathbf{q}$ on surface such that $dist(\mathbf{p}, \mathbf{q})$ is minimal.

$$dist_S(\mathbf{p}) = \min_{\mathbf{q} \in S} dist(\mathbf{p}, \mathbf{q})$$

The finding of the closest point $\mathbf{q}$ in euclidean metric space may be too expensive operation.

Algebraic Distance. The *algebraic distance* is an approximation to Euclidean distance. The definition

$$dist_S(\mathbf{p}) = F(\mathbf{p})$$

satisfy the conditions for metric space. It uses the evaluation of implicit function F of surface S. The value of $F(\mathbf{p})$ is increasing for the points near surface as we go away from the surface. In our implementation we use this distance.

Taubin's Distance. It is modified algebraic distance that is divided by size gradient vector:

$$dist_S(\mathbf{p}) = \frac{F(\mathbf{p})}{|\nabla F(\mathbf{p})|}$$

It suppresses the effect of uneven speed of growth of distance. It is a first order approximation to the euclidean distance.

To find the function that is relatively close to all sample points we define *objective function* which is also called the error function. The objective function of surface S given by parameters $\{c_i\}_{i\in I}$ to set of points P and usually has the form:

$$\sum_{\mathbf{p_i}\in P} (dist_S(\mathbf{p_i}))^2$$

It expresses the cost of approximation summing the error of each point. Consider a shape of unknown parameters c_i. To find them we must minimize the objective function. We get the approximation to the discrete sample of points.

3.4.3 Normalization

When we are fitting an implicit polynomial it can degenerate to constant zero $F = 0$, then all parameters are zero. We solve this problem by saying that implicit polynomial must be a nonzero point in the parameter space. We call this method as *normalization*. We know various types of normalization. We give two classes of constraints.

- Linear constraints – lead to simple least squares or finding eigenvalues.
- Quadratic constraints – lead to solving of generalized eigenvalue system.

There exists many methods how to manage the normalization. Some of them are described in [Zha97]. It also depends on the definition of distance function, for example in division by gradient must be used an iterative technique.

The objective function can contain additional functionals that form surface to desired shape. An example is thin-plate energy used in RBF (section 2.3.2) expressing of the smoothness of surface. Another example is the normalization by given normals at points used in [JF02].

Here are some examples of normalization for quadric of form $F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2exz + 2fyz + 2gx + 2hy + 2iz + j = 0$:

1. $a + b + c = 1$
2. $a^2 + b^2 + c^2 + d^2 + e^2 + f^2 + g^2 + h^2 + i^2 + j^2 = 1$
3. $j = 1$

We notice that using these normalizations we restrain the parameter space and define new shape – the subclass of general quadric.

Solving the minimization problem $\sum_{i=1}^{N} F(x_i, y_i, z_i)^2$ can be done by least squares fitting by *pseudo inverse technique* in first case and third case. In the second case the solution can be found using eigenvalue finding.

3.4.4 Algebraic Surface Fitting

The purpose of local approximation is to capture local shape of the surface. It is always done by minimizing some energy functional and finding the best parameters for describing reconstructed surface. In general it takes a set of points from relatively small area. We can consider also the weights for each point. To make the approximation reliable the number of selected points in set should be at least N_{min}. We set this value experimentally to $N_{min} = 15$. The algebraic surface fitting method should be simple enough to allow us to construct effective and fast algorithm.

In the next section there are some classes of algebraic surfaces.

Algebraic Surface Classes. Algebraic surface is a surface defined by an equation $f = 0$ where $f : \mathcal{R}^3 \to \mathcal{R}$ is a polynomial.

Tensor-product algebraic B-spline surface. The tensor-product parametric surfaces can be generalized to implicit surfaces by adding third basis function. Tensor product algebraic B-spline surface of order (m_1, m_2, m_3) with uniformly distributed knot vector is

$$Q(x, y, z) = \sum_{(i_1,i_2,i_3)\in I} c_{i_1,i_2,i_3}\, B_{i_1}^{m_1}(x)\, B_{i_2}^{m_2}(y)\, B_{i_3}^{m_3}(z), \quad x, y, z \in [0, 1]$$

where $B_i^m(x)$ is i-th B-spline basis function of order m, I is the index set $I = \{(i_1, i_2, i_3)|0 \leq i_1 \leq m_1, 0 \leq i_2 \leq m_2, 0 \leq i_3 \leq m_3\}$ and c_{i_1,i_2,i_3} are constants.

Bajaj's Algebraic Surfaces. Algebraic surfaces are defined on the domain simplex, in 2D it is a triangle, in 3D it is a tetrahedron. Their definition and properties can be found in [XBX00, XBC00, BX01].

Notation for fitting of various shapes. To define the problem exactly we give common formal definitions. As an input we have set of N points $\mathbf{p_i} \in \mathcal{R}^3$, associated normals $\vec{n}_i \in \mathcal{R}^3$ and weights $w_i \in \mathcal{R}$ for $i \in \{1, \ldots, N\}$. To express the coordinates of point $\mathbf{p}$ or we use following notation $\mathbf{p} = (p_x, p_y, p_z)$. The coordinates of normal $\vec{n}$ are expressed similarly $\vec{n} = (n_x, n_y, n_z)$.

The symbol $||\cdot||$ means the size of vector in Euclidean space and it is computed as follows $||\mathbf{p}|| = \sqrt{p_x^2 + p_y^2 + p_z^2}$.

3.4.5 Plane Fitting

The task is for input points and normals to compute an approximating plane. To define the implicit function of the plane we use dot product of reference point b and the normal

n of the plane:

$$F_{\vec{n},\mathbf{b}}(\mathbf{p}) = (\mathbf{p} - \mathbf{b}) \cdot \vec{n} = 0,$$

where the normal $\vec{n}$ and the base point $\mathbf{b}$ (the one that lies on the surface) are the unknown parameters of this function that have to be estimated.

Simple Estimation. The simplest estimation of the plane can be done by averaging the input points and normals:

$$\mathbf{b} = \frac{\sum_{i=1}^{n} w_i \, \mathbf{p_i}}{\sum_{i=1}^{n} w_i}, \; \vec{n} = \frac{\sum_{i=1}^{n} w_i \, \vec{n}_i}{\sum_{i=1}^{n} w_i}.$$

Least Squares Method. This version uses the least squares fitting of points and also normals with weights. To make the equations simpler we rewrite the normal $\vec{n} = [a, b, c]$ and the constant factor to d. We get implicit function of plane in this form:

$$F_{a,b,c,d}(x, y, z) = a\,x + b\,y + c\,z - d = 0$$

The derivatives in all directions are:

$$\frac{\partial F(x,y,z)}{\partial x} = a, \; \frac{\partial F(x,y,z)}{\partial y} = b, \; \frac{\partial F(x,y,z)}{\partial z} = c.$$

To compute the coefficients for approximation we have to find the minimum of this objective function:

$$f_{obj}(a, b, c, d) = \sum_{i=1}^{N} w_i \, F(\mathbf{p_i})^2 + W \sum_{i=1}^{N} w_i \; || \bigtriangledown F(\mathbf{p_i}) - \vec{n}_i | \, |^2,$$

where variable $W > 0$ is used to adjust the weight of normals, the default value is 1.

We solve it as the linear least squares problem. Then all partial derivatives $\frac{\partial f_{obj}}{\partial a}$, $\frac{\partial f_{obj}}{\partial b}$, $\frac{\partial f_{obj}}{\partial c}$, $\frac{\partial f_{obj}}{\partial d}$ should be equal to 0.

$$\frac{\partial f_{obj}}{\partial a} = \sum_{i=1}^{N} w_i \, 2 \, \Big(p_{i_x} a + p_{i_y} b + p_{i_z} c - d\Big) \, p_{i_x} + W \sum_{i=1}^{N} w_i \, 2 \, (a - n_{i_x}) = 0,$$

$$\frac{\partial f_{obj}}{\partial b} = \sum_{i=1}^{N} w_i \, 2 \, \Big(p_{i_x} a + p_{i_y} b + p_{i_z} c - d\Big) \, p_{i_y} + W \sum_{i=1}^{N} w_i \, 2 \, \Big(b - n_{i_y}\Big) = 0,$$

$$\frac{\partial f_{obj}}{\partial c} = \sum_{i=1}^{N} w_i \, 2 \, \Big(p_{i_x} a + p_{i_y} b + p_{i_z} c - d\Big) \, p_{i_z} + W \sum_{i=1}^{N} w_i \, 2 \, (c - n_{i_z}) = 0,$$

$$\frac{\partial f_{obj}}{\partial d} = \sum_{i=1}^{N} w_i \, 2 \, \Big(p_{i_x} a + p_{i_y} b + p_{i_z} c - d\Big) \, (-1) = 0.$$

From these 4 equations with 4 unknowns a, b, c, d we get following linear system:

$$A\,X = B,$$

where

$$A = \begin{bmatrix} \sum_i w_i\,(p_{ix}^2 + W) & \sum_i w_i\,p_{ix}\,p_{iy} & \sum_i w_i\,p_{ix}\,p_{iz} & \sum_i w_i\,p_{ix} \\ \sum_i w_i\,p_{ix}\,p_{iy} & \sum_i w_i\,\left(p_{iy}^2 + W\right) & \sum_i w_i\,p_{iy}\,p_{iz} & \sum_i w_i\,p_{iy} \\ \sum_i w_i\,p_{ix}\,p_{iz} & \sum_i w_i\,p_{iy}\,p_{iz} & \sum_i w_i\,(p_{iz}^2 + W) & \sum_i w_i\,p_{iz} \\ \sum_i w_i\,p_{ix} & \sum_i w_i\,p_{iy} & \sum_i w_i\,p_{iz} & \sum_i w_i \end{bmatrix},$$

$$B = \begin{bmatrix} \sum_i w_i\,n_{ix} & \sum_i w_i\,n_{iy} & \sum_i w_i\,n_{iz} & 0 \end{bmatrix}^T,$$

$$X = \begin{bmatrix} a & b & c & -d \end{bmatrix}^T.$$

The matrix A is symmetric and positive definite, and all normals are of unit size. To solve it we use the Cholesky decomposition to lower and upper diagonal matrix $A = L\,L^T$. First we solve $L\,Y = B$ by back substitution, then from $L^T\,X = Y$ we obtain X (again by backsubstitution). The brief description of Cholesky decomposition and the algorithm can be found in Numerical Recipes [PTVF92].

3.4.6 General Quadric Fitting

The task is to compute an implicit quadratic surface approximating the input points and normals with weights as it was stated earlier. We use following equation for the quadratic implicit surface function:

$$F_A(p) = a_0\,p_x^2 + a_1\,p_x\,p_y + a_2\,p_y^2 + a_3\,p_x\,p_y + a_4\,p_x\,p_z + a_5\,p_y\,p_z + a_6\,p_x + a_7\,p_y + a_8\,p_z + a_9 = 0,$$

where the normal $A = [a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9]$ is the unknown vector of parameters.

The derivatives of F in all directions are:

$$\frac{\partial F(\mathbf{p})}{\partial x} = a_0\,2\,p_x + a_3\,p_y + a_4\,p_z + a_6,$$

$$\frac{\partial F(\mathbf{p})}{\partial y} = a_1\,2\,p_y + a_3\,p_y + a_5\,p_z + a_7,$$

$$\frac{\partial F(\mathbf{p})}{\partial z} = a_2\,2\,p_z + a_4\,p_y + a_5\,p_z + a_8,$$

To compute the coefficients we need to find the minimum of this objective function:

$$f_{obj}(A) = \sum_{i=1}^{N} w_i\,F(\mathbf{p_i})^2 + W\sum_{i=1}^{N} w_i\,\left|\bigtriangledown F(\mathbf{p_i}) - \vec{n}_i\right|^2,$$

where variable $W > 0$ is used to adjust the weight of normals, the default value is 1.

We solve it as the linear least squares problem using the same technique as it was in previous section. Finally we get a symmetric positive definite matrix A. To solve it we use again the Cholesky decomposition.

3.4.7 Bivariate Quadric Fitting

It is a function that returns height for some 2D coordinate.

$$F(u, v) = a_0\, u^2 + a_1\, uv + a_2\, v^2 + a_3\, u + a_4\, v + a_5$$

To transform to 3D we need to define the local coordinate system. We do it by fixing a point, and two base direction vectors $\vec{a}$ and $\vec{b}$. These vectors are of unit size. Then the evaluation an implicit function is following. For a point $\mathbf{p} = [x, y, z]$ firstly the $[u, v]$ coordinates is computed:

$$u = \mathbf{p} \cdot \vec{a} = p_x\, a_x + p_y\, a_y + p_z\, a_z,$$

$$v = \mathbf{p} \cdot \vec{b} = p_x\, b_x + p_y\, b_y + p_z\, b_z$$

$$w = \mathbf{p} \cdot \vec{n} = p_x\, n_x + p_y\, n_y + p_z\, n_z$$

Then the function $F(u, v)$ is evaluated as follows: $F(x, y, z) = F_{\vec{a},\vec{b}}(u, v) - g$.

We compute the approximation similarly we have done it in two previous sections. The difference is that we do not use normals because it is not necessary to normalize the surface, it cannot degenerate. We use our local coordinate system defined by $\vec{a}$, $\vec{b}$ and $\vec{n}$ and each point $\mathbf{p_i}$ is transformed to u_i, v_i and g_i using the formulas above. We find the minimum of this objective function:

$$f_{obj}(A) = \sum_{i=1}^{N} w_i \left(F_A(u_i, v_i) - g_i\right)^2$$

where variable A is the unknown vector of coefficients.

Finally we get the linear system $A \cdot X = B$ with symmetric matrix A that can be solved using Cholesky decomposition, again.

Note about this shape class. It avoids artifacts because this kind of surface always only consists of one segment continuous and connected.

3.4.8 Ohtake's Fitting of Quadric

Solution we use was described in [OBA+03] and it improves fitting of general quadric. It uses extraordinary points placed in the space out of surface. We are given a are set of

candidate positions $\{\mathbf{q_j}\}_j$. For each point $\mathbf{q_j}$, 6 nearest points $\mathbf{p_{ji}}$ is collected. If not all 6 associated normals have same orientation ($\vec{n}_{ji_1} \cdot \vec{n}_{ji_2} < 0$, for all $i_1 \neq i_2$), then the point $\mathbf{q_j}$ is removed. For all remaining points (let us call them $\{\mathbf{r_1}, \ldots, \mathbf{r_m}\} \subset \{\mathbf{q_j}\}_j$) the estimated distance from surface is computed

$$d(\mathbf{r_j}) = \frac{1}{m} \sum_{i=1}^{m} \vec{n}_{ji} \cdot (\mathbf{r_j} - \mathbf{p_{ji}}) .$$

To compute the coefficients we need to find the minimum of this objective function:

$$f_{obj}(A) = \frac{1}{\sum w_i} \sum_{i=1}^{N} w_i \, F(\mathbf{p_i})^2 + \frac{1}{m} \sum_{j=1}^{m} \left(F(\mathbf{p_j}) - d(\mathbf{r_j})\right)^2 ,$$

and find parameters describing $F(\mathbf{x})$.

Note about this shape class. It avoids artifacts such as unwanted branches very good. During the testing it there were no such artifacts generated.

3.5 Blending

The blending in general is used to achieve smooth transition of some objects, said in mathematical language, functions. The purpose of surface blending is to achieve smooth final surface using a mathematical formula. We are given two or more pieces of surfaces that may be disconnected. The level of smoothness is measured in the degree of continuity. The degree of continuity depends on used blending functions and also on blended functions. The final degree of continuity in one point of space p is equal to minimal degree of all used functions f.

The blending functions can be divided into two main categories according to their definition scope – the support.

- Compactly Supported – they have nonzero value only on the finite bounded space. This class allows to construct fast reconstruction algorithm and fast evaluation algorithm too because of their compactness but we need to set proper radius of support more carefully (to preserve the approximative uniform covering).

- Uncompactly Supported – they can be evaluated to nonzero value in any point of infinite space. The whole space where they are nonzero is infinite. This class of blending functions have good extrapolation properties and therefore they are suitable for reconstruction of larger holes.

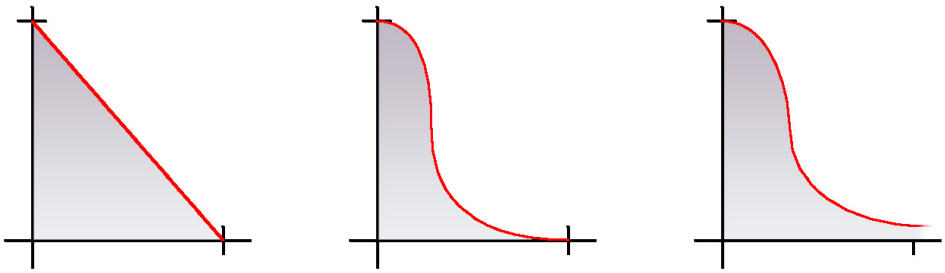


Figure 3.5: Blending Functions

There is another important property of blending functions. If we compare Bernstein-Bézier or B-spline blending functions to rational blending, the first group has a nice property that the blended result cannot exceed the convex hull of input arguments. If we use rational blending function then the result may slightly exceed and an unnatural curvature is created.

In spite of this fact, we use rational blending functions because they allows easy construction of result function instead the local approximations are not placed in regular shape (such as uniform grid). If we want to blend n local approximating functions $f_i : \mathcal{R}^3 \rightarrow \mathcal{R}$ with associated weighting functions $w_i : \mathcal{R}^3 \rightarrow \mathcal{R}$, then we use this formula:

$$f(\mathbf{x}) = \frac{\sum_i w_i(\mathbf{x}) f_i(\mathbf{x})}{\sum_i w_i(\mathbf{x})}$$

where the evaluation of $f(\mathbf{x})$ is usually centered to c_i and the weighting functions usually have adjustable radius ($r > 0$). Here are some variants of blending functions: (figures 3.5)

- Linear form:
$$w(||\mathbf{x} - \mathbf{c_i}||) = \begin{cases} (1 - \frac{||\mathbf{x}-\mathbf{c_i}||}{r}), & if\ ||\mathbf{x} - \mathbf{c_i}|| < r \\ 0, & otherwise \end{cases}$$
This function is compactly supported of continuity of degree 0.

- Piecewise quadratic form that is derived from B-spline basis function of second order:
$$w(||\mathbf{x} - \mathbf{c_i}||) = \begin{cases} \frac{3}{4}\left(1 - 3\left(\frac{||\mathbf{x}-\mathbf{c_i}||}{r}\right)^2\right), & if\ ||\mathbf{x} - \mathbf{c_i}|| \leq \frac{r}{3} \\ \frac{9}{8}\left(1 - \left(\frac{||\mathbf{x}-\mathbf{c_i}||}{r}\right)\right), & if\ \frac{r}{3} < ||\mathbf{x} - \mathbf{c_i}|| \leq r \\ 0, & if\ ||\mathbf{x} - \mathbf{c_i}|| > r \end{cases}$$
This function is compactly supported of continuity of degree 1.

- Exponential form of gauss curve used for normal distribution:

$$w(||\mathbf{x} - \mathbf{c_i}||) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{||\mathbf{x}-\mathbf{c_i}||^2}{2\sigma^2}}$$

  This function is uncompactly supported of infinite continuity. For this function the parameter variance σ^2 is adjustable.

Use of basis function with compact support preserves the principle of locality and therefore it is suitable to use it with local approximations. All blending methods should hold that there is no difference in blending of different local approximations over space. The blending parameters is allowed to change only depending on the geometry of the cell of local approximation.

The representation we choose depends on blending we are using. If we choose the partition of unity blending, then we do this assignment by defining sphere of effect. This sphere is defined by its center and radius. Other parameters for blending functions can be included.

RBF blending. In our algorithm we use the octree subdivision. RBF blending [CBC+01] uses spherical compact support. For each local approximation $f_i(\mathbf{x})$, a center $\mathbf{c_i}$ and radius r_i is assigned. Our algorithm places the centers to the leaf of octree.

We notice that this approach places the centers approximately uniformly distributed. Therefore it is one of the heuristic approximations to Voronoi diagram (section 3.2.4).

3.6 Multi-level Partition of Unity Implicits

In this section we give detailed description of used algorithm. It based on [OBA+03] with some implementation improvements. We write about the mathematical model of implicit surface, the possible generalization of the model and the modifications we can create. At the end we add our comments on this method and we refer to results that can be found chapter 5.

We have already described some parts of our algorithm. Local surface fitting is in section 3.4.4, the sampling requirements has been described in section 3.3, estimation of normals are described in next chapter 3.7, octree subdivision technique is in section 3.2.3. We use these references in next description of MPUI algorithm.

Iterative Model. We use build the reconstructed surface iteratively. We start with first approximation $F_0 : \mathcal{R}^3 \to \mathcal{R}$ of general quadric to the set of points $P = \{p_i\}_i$. Then we

recursively refine the surface F_i in the places where the error of approximation is large and we create a new approximation F_{i+1}.

$$F_0 \rightarrow F_1 \rightarrow \ldots \rightarrow F_k$$

We use octree subdivision scheme so we replace the node N with eight new nodes $N_1, \ldots, N_8$. We continue with refining until the error of approximation is below the desired level. This process converges to the implicit surface $F_k : \mathcal{R}^3 \rightarrow \mathcal{R}$ when the input points P are sampled from 2-manifold object. In practice we restrict the maximum level recursion.

3.6.1 Local Approximations

For local approximations we use following shape classes: general quadric, bivariate quadric and piecewise quadric for sharp features. We use only second degree polynomials because it is enough to capture the curvatures which are noticeable to human eyes. It is usual in geometrical modeling to use piecewise quadratic polynomials.

To make local approximation reliable we set the minimum required count of points $N_{min} = 15$ for computation of local approximation. If there are not enough points the radius is iteratively increased (look in section 3.6.3).

The local shape function is chosen according to analysis of points and normals. If the number of points in current cell is large ($|\mathcal{P}| > 2N_{min}$) then one of the quadric is fitted. If the number $|\mathcal{P}| \leq 2N_{min}$, then the set of points is tested for the presence of sharp feature.

Error measurement. For the measurement of error we use the modified Taubin distance [Tau91]

$$\varepsilon = max_{||\mathbf{p_i}-\mathbf{c}||<r} \frac{F(\mathbf{p_i})}{||\nabla F(\mathbf{p_i})||},$$

where $\mathbf{p_i}$ are the sample points, $\mathbf{c}$ is center of cell, r is radius of support sphere, and F is implicit function of local approximation.

General Quadric. It is for capturing of general shapes like parabolic, hyperbolic but also elliptic surfaces. We use this shape class for coarse approximations in lower levels of subdivision tree.

For this shape class is characteristic that the maximum deviation of normals θ is large. Therefore the condition for fitting

$$|\mathcal{P}| > 2N_{min} \quad \text{and} \quad \theta \geq \frac{\pi}{2}.$$

includes the test for this angle. The fitting of this shape class was described in section 3.4.8.

For better result of fitting we use the improved technique supposed by [OBA+03]. He uses 8 auxiliary points away placed away from surface, in the corners of current cube cell. In these points the estimated distance was approximated so the resulting surface was more relevant and the number of unwanted branches of implicit surface was reduced to minimum. During our testing of this method no such artifact was created.

Bivariate Quadric. It is for capturing local shape with simple topology that can be projected to plane and the local shape function is the function of height in local coordinates.

This shape class consists of one segment. We suppose that the maximum deviation of normals θ is not big. The conditions for fitting this shape class are

$$|\mathcal{P}| > 2N_{min} \quad \text{and} \quad \theta < \frac{\pi}{2}.$$

The fitting of this shape class was described in section 3.4.7.

Piecewise Quadric. It is used for sharp features like edges and corners, this is discussed in chapter 4.

This shape class is used only for set with fewer points that N_{min}

$$|\mathcal{P}| \leq 2N_{min},$$

and the test for features presence was successful. The tests are described in detail in chapter 4 in sections 4.3.1 and 4.3.3. If no conditions for sharp feature is satisfied then bivariate local shape is used.

3.6.2 Description of Algorithm

In this section we describe the algorithm with help of the pseudo-code in the figure 3.6 and we discuss the properties of algorithm implied by our construction.

Parameters. The algorithm evaluates the implicit function in a *point* by traversing all relevant cubes in octree. Except the point there are two other parameters: *max_error* and *max_depth*. They are used for condition for stopping subdividing of current cell (figure 3.6, function Cell.MPU_approx(), line 5). Subdivision continues until error of approximation in current cell is more that *max_error* and the depth of recursion is less than *maxDepth*.

Compact evaluation. The algorithm traverses only relevant cells. The cell is relevant if the evaluation *point* is in the support sphere of the cell. If the point is out of this sphere, then the cell can be missed in evaluation (figure 3.6, function Cell.MPU_approx(), line 1). The support sphere is computed in local approximation.

```
Evaluate_MPU_approx(point, max_error, max_depth)
1:    sum_values = 0;
2:    sum_weight = 0;
3:    root.MPU_approx(point, max_error, max_depth, sum_values, sum_weight);
4:    return (sum_values / sum_weight);

Cell.MPU_approx(point, max_error, max_depth, sum_values, sum_weight)
1:    if (|point - this.center| > this.radius) then
2:        return;
3:    if (this.local_approx is not computed yet) then
4:        compute this.local_approx;
5:    if (this.error > max_error) and (this.depth <= max_depth) then
6:        if (children not created) then
7:            create 8 children;
8:        for each child do
9:            child.MPU_approx(point, max_error, sum_values, sum_weight);
10:   else
11:       sum_values += this.weight(point) * this.local_approx(point);
12:       sum_weight += this.weight(point);
```

Figure 3.6: Pseudo-code of Algorithm

Rational evaluation. The algorithm compute sum up all evaluations in variable *sum_values* and all weights in variable *sum_weight* in leafs of octree (figure 3.6, function Cell.MPU_approx(), line 11-12). The resulting value is computed as the total sum divided by total weight (figure 3.6, function Evaluate_MPU_approx(), line 4).

Evaluation on demand. Another important property of this algorithm is that the octree is built on demand. The tree can be built only partly, the cell is subdivided (figure 3.6, function Cell.MPU_approx(), line 6-7), when the conditions are satisfied. This can be used for example when we want to ray-trace the image of one side of object, then only visible part of object if reconstructed.

Reuse. Another property is that the octree structure can be reused. The evaluation can be run on the previously created structure with more strictly parameters *max_error* and *max_depth* and the refined reconstruction can be achieved.

Locality. The design of this algorithm preserves the locality principle of local approximations and blending. It is optimized to speed of reconstruction and evaluation. The

algorithm use octree subdivision that is the simplest subdivision scheme. The computing of local approximations is optimized, and the evaluation of global implicit surface is performed on demand.

3.6.3 Blending

We have decided to use RBF blending with radial support that was described in section 3.5. We use octree subdivision which is usually considered with cube cells but we use sphere cells. The overlapping of spheres makes no problems because the rational blending can manage any count of blended functions.

Moreover we use that support spheres can overlap and set the larger radius than diagonal of cube $\frac{\sqrt{3}}{2}$. We use 1.5 of this value so the radius is equal to

$$r = \frac{3\sqrt{3}}{4} \cdot d,$$

where d is the size of side of cell cube;

In the section we have defined the minimum number of points in the support sphere N_{min}. If the support sphere does not contain enough points, then the sphere is increased by multiplication.

$$R = R\,(1 + \lambda)\,,$$

where $\lambda > 0$ and usually $\lambda < 0.5$, we use $\lambda = 0.2$.

3.7 Normals Estimation

Description of algorithm of orientation propagation based on Kruskal's minimum spanning tree finding. Start from one point where we are sure with the orientation of normal. Expand the set of correctly oriented normals using MST. For each adding of edge we check if orientation of two groups is synchronized. If not then swap one of the group. (figure 3.7)

Algorithm. Normal estimation is done locally by linear regression plane by least-squares fitting of plane to set of points.

The error of orientation can occur, that is that the direction of estimated normal is correct but the orientation is opposite (figure 3.7, left).

We need to synchronize the orientation of all normals and we use the information about neighboors. Two neighbooring points $\mathbf{p}$ and $\mathbf{q}$ with normals $\vec{n}_p$ and $\vec{n}_q$ have synchronized orientation of normals if

$$\vec{n}_p \cdot \vec{n}_q > 0.$$

Figure 3.7: Normals Orientation Propagation

In the construction of algorithm we use the fact that we have the points are well sampled from surface and the deviation of directions of normals is not very large for neighboring points.

The algorithm considers the set of correctly oriented normals O, and it propagate the orientation of this set in this way. It adds the point that is not in set and it is one of the closest neighbors to set S. Each time the new point $\mathbf{p}$ is processed the orientation of this point is compared with the closest point in set O. If the orientation is opposite, then the normal $\vec{n}_p$ is inverted. Finally the point $\mathbf{p}$ is added to the set O.

The analysis of this algorithm leads us to the observation that it works like algorithm for finding minimum spanning tree (MST). The fashion of expanding is equal to construction of MST for weighted graph where vertexes are points and the weight of edge between two points $\mathbf{p}$, $\mathbf{q}$ is:

$$weight(E(\mathbf{p}, \mathbf{q})) = dist(\mathbf{p}, \mathbf{q}).$$

We can replace the algorithm with the Kruskal's one and we get change of fashion of orientation propagation. Instead of expanding set O like it was in previous case, we observe that more clusters are expanding concurrently. Each time when two clusters are merged, two closest points $\mathbf{p}$ and $\mathbf{q}$ from neighboring sets O_1 and O_2 are checked if their orientation is synchronized. If this is not true, then all normals in one group (we take that one where are fewer points) must be inverted.

This algorithm can be used also for analyzing the count of connected segments, if we set the threshold $maxdist$ for maximum distance of two points. In language of graph theory, we remove all edges with value more than $maxdist$. We can count the segments during this algorithm. The normals in result set are synchronized in each segment independently and therefore we avoided the errors of synchronizing orientation between two distant segments.

The quality of result of this algorithm can be improved by setting of the weights of graph considering also the dot product of normals. The weight of edge incident to two points p and q with normals $\vec{n}_p$ and $\vec{n}_q$ is set as follows:

$$weight(E(\mathbf{p}, \mathbf{q})) = \frac{dist(\mathbf{p}, \mathbf{q})}{|\vec{n}_p \cdot \vec{n}_q|}.$$

This formula increases the weight of points that have large deviation of normals. They can be close to each other but if their normals are almost opposite then the weight is too large to connect these two points in MST. Using this improvement made the implementation of normal estimation algorithm more stable.

3.8 Conclusion

We have discussed the problem of surface reconstruction and we have presented a solution that uses implicit surface representation. We have discussed the sampling requirements, various space subdivision techniques, local approximations computing, blending techniques and normals estimation. We have decided to implement the algorithm MPUI as it was described in section 3.6. More about implementation is in appendix A. Results can be found in chapter 5.

Chapter 4

Features Reconstruction

4.1 Introduction

The notion of feature is very common and it is used in many areas. We focus only on the features in geometric modeling. The notion of feature in geometry is connected with the notion of shape. The features are important for visual perception and recognition of object. We can say that feature is anything that characterizes the shape of object.

Philosophy. Feature in general is some part of object that characterize object. After its removal the object changes the characteristics. This is said under global view. In contrast, the feature under the local view is a local shape and the library of features is the basis of local shapes. The shape of whole object is built using this library. We follow the philosophy of the local view. The common thing for these two views is that the feature is visible under different scale levels.

Perception of Features. Human visual system detects features on low level and it recognizes whole object from them on high level. It can reconstruct incomplete data and it gives the interpretation of recognized data that may be also false. The techniques of numerical approximation give sufficient result for a given error measurement. Our task is to make efficient model for reconstruction of surface with sharp features that is plausible not only for the error measurement but also for human visual system.

Modeling of features. We have recognized these steps in feature reconstruction:

1. Detection – testing if any feature is present in current set of data.
2. Reconstruction – identifying the segments and modeling the feature.

The detailed description of these two steps can be found in following sections.

Our Direction. We deal only with the sharp features any sharp local shape such as edges, corners. The sharp feature can be detected as a singular point or set of points. It cannot be approximated by any finite sequence of polynomial functions. There is no match with Weierstrass theorem mentioned in section 3.4.

Assumptions. We assume that the points are uniformly sampled over the whole surface from small area and therefore there are no big holes. The segments are almost linear, because the points are from small area. They can be curved but not very much so that bivariate polynomial can be fitted to them. The important assumption is that we detect only sharp features, where we are sure that it is a feature.

4.2 Various Approaches

We have identified three approaches for detection and reconstruction of features. All of them analyze the input set of points and normals but in different way. We give only basic idea of each of them.

Kobbelt's Heuristic Approach. Kobbelt et al. [KBSS] use a heuristic approach. He computes the maximum deviation θ of any two normals. If it is more than specified threshold (he use $\theta = \frac{\pi}{2}$), then the feature is detected. Then follows the construction of segments. This approach is described in section 4.3.2.

Principal Component Analysis. Pauly [Pau03, p. 13-14] uses analysis of the principal component vectors q_1, q_2, q_3. The flat surface area has the variance in two vector larger, and in one it is significantly smaller. The feature is detected when this configuration does not hold.

Minimum Spanning Tree. This approach analyzes the weighted graph of points. The weights are created with respect to the distance of points and the deviation of normals. We have used this technique for normals estimation (section 3.7). We will use this technique for features fairing so the detailed description can be found in section 4.4.

4.3 Sharp Features

We deal only with sharp features that can be easily detected as singular values. The implicit surface in such kind of values is continuous but it is not differentiable. We deal with the certain local features as edges and corners.

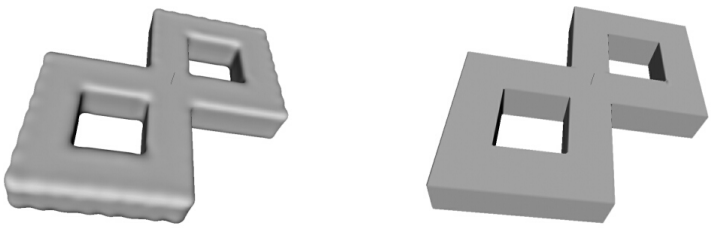


Figure 4.1: Reconstruction without and with Sharp Features.

When we try to construct polynomial approximation on the sharp parts the error is too large. The theory of approximation says that for such singular values cannot be constructed any finite polynomial approximation. We can see an attempt of reconstruction using only polynomials on figure 4.1.

Therefore we propose to extend the model of implicit surfaces by sharp features. This decision makes the problem simpler because we do not have to deal with the topology of features that is often complicated. Using sharp features the count of local approximations rapidly decreases. As we have said, we have recognized two main classes of local features: edges (with two segments intersecting in singular line) and corners (three or more intersecting segments).

4.3.1 Modeling of Features

When we model the sharp features with implicit surfaces, in contrast to parametric surfaces, we do not need to know the exact position of edge but we need to know only the shape of intersecting segments and their mutual position. The mutual position of two segments can be either concave or convex. Depending on this configuration we construct the edge using one of the binary functions $min(a, b)$ (for concave) and $max(a, b)$ (for convex) as it is shown on the figure 4.2. For corner feature we have at least three segments. Except the fully concave and fully convex corner there are lots of saddle cases that must be modeled. The detailed description of the modeling of saddle corner can be found in section 4.3.3

4.3.2 Detection of Features

For the set of points $S = \{\mathbf{p_i}\}$ with assigned normals $\vec{n}_i$ we need to detect the presence of a feature. If the maximum deviation of angle of two normals $\vec{n}_1$ and $\vec{n}_2$ is above specified threshold $\theta = 25°$ (the value is a result of experiments), then we consider it as a feature. Then we are testing if there is another segment by finding a normal $\vec{n}_3$ with maximally deviating angle from both previously detected segment normals $\vec{n}_1$ and $\vec{n}_2$. If this angle is

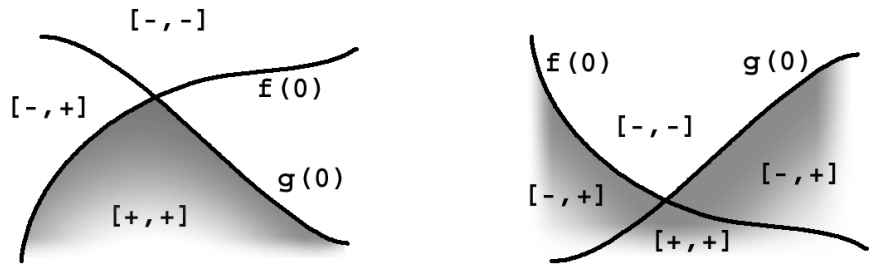


Figure 4.2: Edge Modeling in 2D.

bigger than θ, then we expect the presence of corner. We continue with the same detection for the 4th normal $\vec{n}_4$.

The result of previous segment prediction is the set of normals of k segments. We obtain the partitioning of the set of points S by assigning for each point $\mathbf{p_i}$ one of the segment S_j that has minimum deviation of normal $\vec{n}_i$ to normal $\vec{n}_j$. Then we improve the partitioning by following iterative algorithm. For each segment S_j, we compute new normal $\vec{n}'_j$ as an average of normals in segment. Then we compute again the partitioning S'_j according to newly computed normals. We do this two times and on the end we obtain good estimation of the partitioning to segments. To reduce the error of approximation we remove the points that have deviation from the segment normal greater than 50° (the value is a result of experiments).

4.3.3 Reconstruction of Features

Now we need to construct the implicit surface function for the local feature. For the case with two segments we have already done it in section 4.3.1. We give a technique that allows us to construct corner with any count of segments with the knowledge of convexity of each pair of segments.

The convexity estimation for pair of planes defined by points $\mathbf{p_1}$, $\mathbf{p_2}$ and normals $\vec{n}_1$, $\vec{n}_2$ can be done by evaluation of implicit function of planes. The edge is convex iff $f_1(\mathbf{p_2}) + f_2(\mathbf{p_1}) > 0$, otherwise the edge is concave.

The task is for k segments $F_{1\ldots k}$ and convexity defined for each pair $\text{conv}[F_i, F_j] = \{1, 0\}$ to construct an implicit function $F(x)$ modeling the shape of local feature that satisfy the conditions:

$$\text{conv}[i,j] \Rightarrow F(x) \leq min(F_i(x), F_j(x))$$

$$\neg conv[i,j] \Rightarrow F(x) \geq min(F_i(x), F_j(x))$$

The function $F(x)$ can be constructed by collecting all convex pairs $conv[F_i, F_j]$:

$$F = min(max(F_{i_1}, F_{j_1}), max(F_{i_2}, F_{j_2}), \ldots)$$

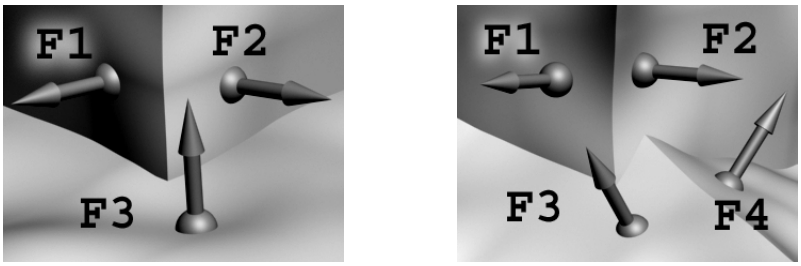


Figure 4.3: Modeling of Settle Corners.

The conditions are satisfied because the *min* and *max* function are monotonous in structure addition:

$$min(\ldots F_i \ldots F_j \ldots) \leq min(F_i, F_j),$$

$$max(\ldots F_i \ldots F_j \ldots) \geq max(F_i, F_j),$$

The expression constructing function F can be optimized using DeMorgan rules:

$$min(A, max(B, C)) = max(min(A, B), min(A, C)),$$

$$max(A, min(B, C)) = min(max(A, B), max(A, C)),$$

because the algebraic structure $(F(x), min, max)$ is an generalization of Boolean algebra.

On figure 4.3 on left the settle corner with the set of conditions $conv[F_1, F_2]$, $\neg conv[F_1, F_3]$, $\neg conv[F_2, F_3]$ is modeled as the function $F(x) = max(min(F_1, F_2), F_3)$.

On figure 4.3 on right the settle corner with the set of 2 convex edges $conv[F_1, F_2]$, $conv[F_3, F_4]$ is modeled as the function $F(x) = max(min(F_1, F_2), min(F_3, F_4))$.

4.4 Features Fairing

We have constructed an algorithm for local feature reconstruction so we are able to reconstruct some features on real objects. By analysis of results of previous algorithm we obtain two different groups of objects:

- In the first group there are objects for civil engineering industry, CAGD models. They have large areas with low curvature, and sharp strong edges. The features on the objects of this class can be easily detected because they are sparsely placed.

- In the second group there are real objects that can come from 3D scanner. The objects of this class have large amount of places with high curvature and densely placed features. That is the reason why the topology of features cannot be easily resolved.

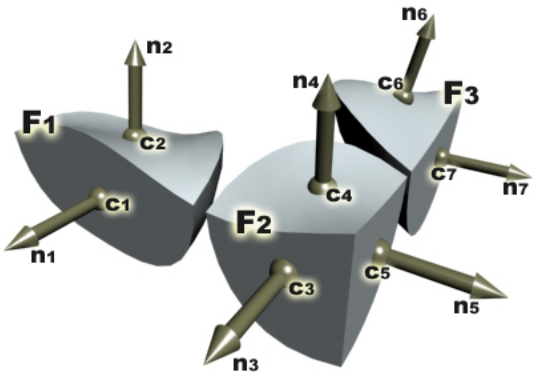


Figure 4.4: Standard Topology of Features

The features fairing on the objects in the second class is more difficult and more complex and the solution needs to use not only sharp local features. Our primary purpose is to construct features on the objects from first class.

Topology of Features. We need to distinguish the correct configuration of features from the bad one. For checking it in our case, it can be used the information about the topology of features (the adjacency of features). The typical topology of features for corner case is that adjacent features are edges (figure 4.4). Our goal is to check the correctness of the topology of features.

Purpose of Features Fairings. There are at least two main purposes of feature fairing.

- The first one is the improvement the quality of feature considering the topology of features, in general we say we do a filtering the features from noise and from bad results from previous step of local reconstruction.
- The second purpose is to repair some bad considerations that were done in previous step. Specifically we have done convexity estimation for local feature construction (section 4.3.3). This convexity estimation can be corrected according to adjacent features.

Typical Errors. The task is to resolve these typical error configurations:

- There is an isolated feature with no adjacent features. Such feature should be removed.

- Edge feature with one error, (all features have two segments, but only one is detected as corner or there is no feature). This configuration must be detected as the error. In the next step the new feature is computed regarding the adjacent segments in features.
- There are 3 adjacent edges and the corner was not detected. This configuration should be detected and repaired.

4.4.1 Technique

The algorithm for features fairing must detect above described errors and replace them by newly computed sharp feature. The main principle is that we use larger neighborhood than it was in local feature reconstruction. We use the support sphere with double radius $2r$ for fairing of this feature.

Instead of analyzing the local shape functions we analyze the segments that were used to model the feature. For the edge we have two segments, for the corner we have at least three segments. Each segment is a bivariate polynomial so we expect the segments are nearly planar. Therefore we approximate each segment s by one point $\mathbf{c}$ and one normal $\vec{n}$ defining the plane.

Weighted Graph. For purposes of analysis of the input set of local features we define the graph structure G that express the relationship between segments. The graph $G = (V_G, E_G)$ consists of segments $V_G = \{s_i = (\mathbf{c_i}, \vec{n}_i)\}$ with weighted edges $E_G = \{e_k = (s_i, s_j)\}$. The weights must include the distance of two points $\mathbf{c_i}$ and $\mathbf{c_j}$ and the deviation of two normals $\vec{n}_i$ and $\vec{n}_j$. We propose this setting of weight w_{ij} of two segments s_i and s_j:

$$w_{ij} = ||\mathbf{p_i} - \mathbf{p_j}||^2 + R\, e^{-\vec{n}_i \vec{n}_j}, \quad R > 0.$$

This setting assigns lower weights to closer points, and lower weights to the normals with less deviation. The parameter R is used to adjust the strength of deviation of normals. We set $R = 1$.

Segmentation. The goal is to compute the number of segments of feature and their position. We do it by analysis of the graph G. We obtain the segmentation of vertexes $V_G = \bigcup_{h=1}^{H} V_h$. Then for each segment V_h we assign a point $\mathbf{c'_k}$ and a normal $\vec{n}'_k$.

Improved MST. We may do the segmentation using any clusterization technique. We propose our technique based on finding minimum spanning tree (MST) that was described in 3.7. We construct the set for minimal spanning forest by adding the edges with minimal weight. The result is disconnected set of MSTs, each of them is a segment V_h.

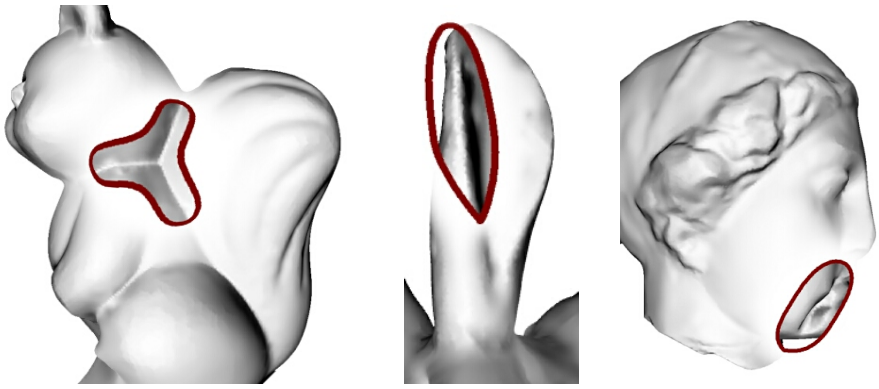


Figure 4.5: Features on Real Objects.

We improve this algorithm by replacing the test for minimal edge connecting two different segments. This test is unstable for certain types of configurations. We replace it by evaluating two such minimal edges. Generally, we connect such two spanning trees t_i and t_j that have minimal value of this expression

$$o(s_i, s_j) = \sum_{e \in J_{ij}} w(e)$$

for set J_{ij} containing the L edges with lowest weight connection and connection two spanning trees t_i and t_j. We set L to at least $L = 2$.

New Feature Computation. The final step is to estimate new local shape function. As first, we reconstruct the bivariate polynomial for each cluster V_h. Then we model the new local function $f : \mathcal{R}^3 \rightarrow \mathcal{R}$ using a technique described in section 4.3.3.

4.5 Conclusion

We have presented the technique for reconstruction of certain local features and a technique for fairing of reconstructed local features and we have successfully applied it to the real objects with features.

Some parts of reconstructed models are visible on figure 4.5.

Future Work. As a further extension can be done global features extraction and building the functional representation that characterize the global shape of object. We could do it by taking already detected local features and try to merge them into the global features.

The global features can be used also for fairing of local features to eliminate some errors during local feature reconstruction.

Chapter 5

Conclusion

5.1 Contribution

For the problem of surface reconstruction from unorganized set of points, we have designed a solution based on [OBA+03]. We have done a survey to recent work in surface reconstruction for various representations (chapter 2). We have chosen the implicit surface representation. We build the implicit surface by blending local polynomial approximations with spherical support. We use the iterative technique to build the implicit surface by refining the approximation using octree subdivision scheme.

In the chapter 3, we have discussed various possibilities for subdivision scheme, for local approximations and for blending. We also give the requirements for sampling and the level of noise. We have designed an original algorithm for normals estimation.

Sharp Features. In the chapter 4, we have considered the problem of sharp features reconstruction. We extend the model of implicit surface by the functions capable to model the sharp edges and corners. We present a technique based on [KBSS] for detection and reconstruction of local features. Then we present an original technique for fairing of local features. This technique is capable to repair certain errors from previously reconstructed features.

5.2 Webpage

We notice that there is a webpage for this master thesis that is currently on the address `http://umarian.szm.sk/thesis/index.html`. In the case that the address would change, you just enter the search query `Implicit Surface Reconstruction Marian Uhercik Master Thesis` to the `http://www.google.com`.

5.3 Future work

Implementation of Features Fairing. We have done the detailed design of the algorithm for features fairing. Our current goal is to implement this algorithm and do the tests.

Testing on Large Input Sets. The construction of algorithm is capable to reconstruct the surface from large ammount of input set. The next goal is to test our algorithm on such data sets.

Exporting the Implicit Surface. The easy evaluation of implicit surface from our library function has been already used for creating the plugin for POV-Ray. For some applications it is not enough to have a black-box function. They need to know the more about implicit function We can export the mathematical description of implicit surface to the Hyperfun modeling language.

Library of Local Shapes. We have used only few types of local approximations. The future work can be done in extending this library of local shapes. The important step is a choice of a proper local shape.

Intelligent System. With help of user interaction we can obtain better results. To reduce the amount of user interaction an intelligent system could be built. User can give some examples and the system could learn the assignment of local input data to local shape in a library.

Global Shape Extraction. We have presented new algorithm for feature reconstruction. Our vision is to construct algorithm for extracting global shape. The result can be stored in functional representation. The algorithm can use the local features and merge them to the global features. Global shape extraction can be examined to extract the global shape description from local features.

5.4 Results

We present the results of surface reconstruction by our algorithm. We use the input set of points with normals from the Ohtake's webpage [Oht].

We have tested the algorithm on a personal computer with operating system Windows XP. We have used two different configurations:

- Celeron 1300 MHz with 384 MB of RAM.

- AMD 2500 with 256 MB of RAM.

All times for reconstruction for following data sets were close to zero or one second. The polygonization can take longer time because it evaluates the implicit surface many times.

Squirrel. We reconstruct the implicit surface from data of squirrel. We use only polynomials with no sharp features extension. It contains 40627 points. All times on both configurations were about 1 second.

We adjust the maximum depth of recursion of octree to present the multilevel reconstruction. We set the maximum desired error to 0.5% of object's bounding box. The result is in graphs on figure 5.1, the figure 5.2 shows the polygonized object.

Knot. We reconstruct the implicit surface from the data of knot. It contains 13006 points. We use only polynomials with no sharp features extension.

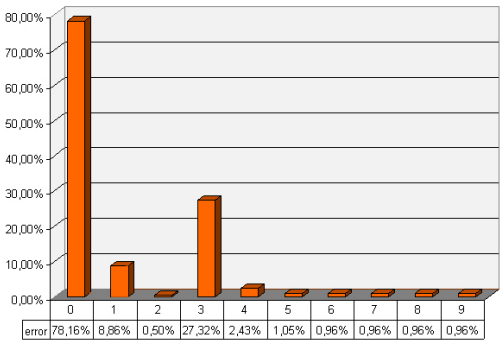
Bunny. We reconstruct the implicit surface from the incomplete data of bunny with many holes. It contains 28060 points. We use only polynomials with no sharp features extension.

Venus. Next reconstruction of the implicit surface is from data of Venus. We use only polynomials with no sharp features extension. It contains 72545 points. All times on both configurations were about 1 second.

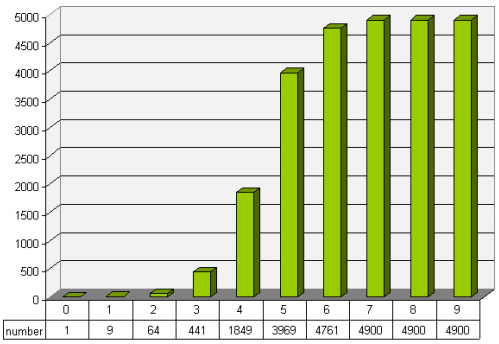
The result o reconstruction can be seen on picture 5.5.

Two-torus. Here are the results of reconstruction of boolean objects with sharp features. The model (figure 5.6) contains 4352 points with normals. To render the pictures we use raytracing technique to preserve the sharp edges.

Boolean objects. Here are the results of reconstruction of boolean objects with sharp features. The first model (figure 5.7) contains 6300 points with normals. The second model (figure figure 5.8) contains 6572 points with normals. To render the pictures we use raytracing technique to preserve the sharp edges.

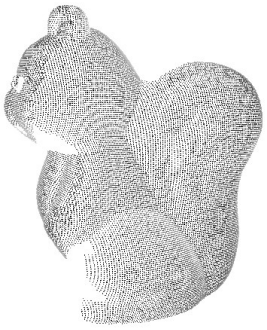


Maximum Taubin's error

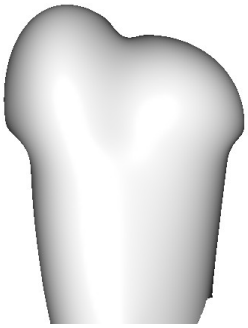


Numbers of local approximations

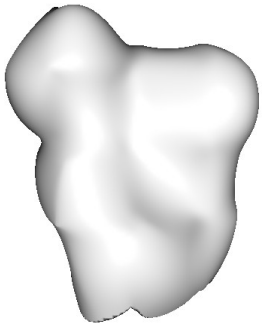
Figure 5.1: Results for Reconstruction of Squirrel Object



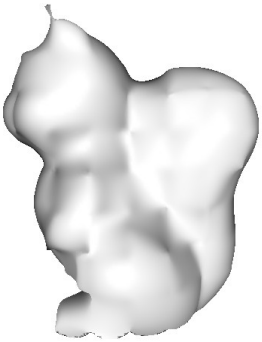
points



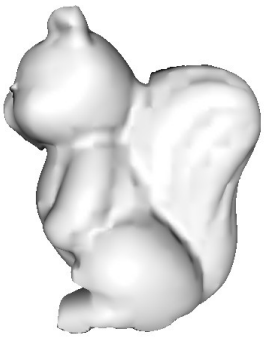
depth 1



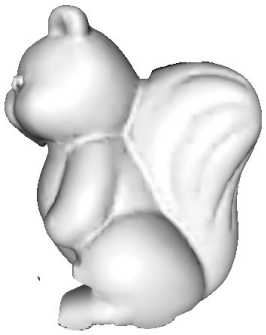
depth 2



depth 3

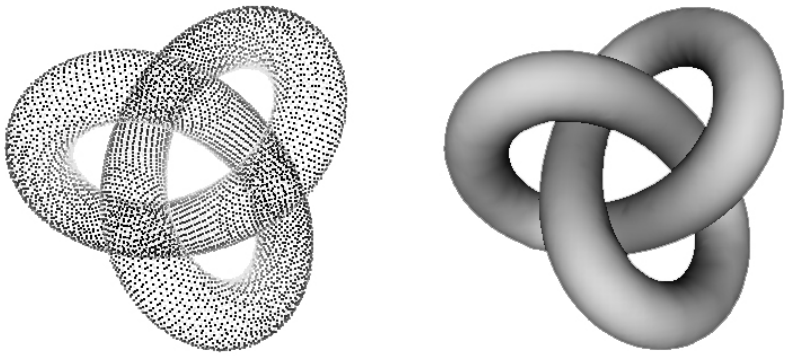


depth 4



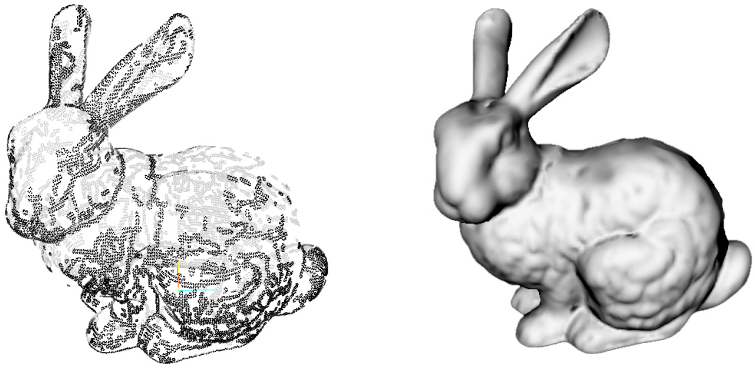
depth 6

Figure 5.2: Multi-level Reconstruction of Squirrel Object



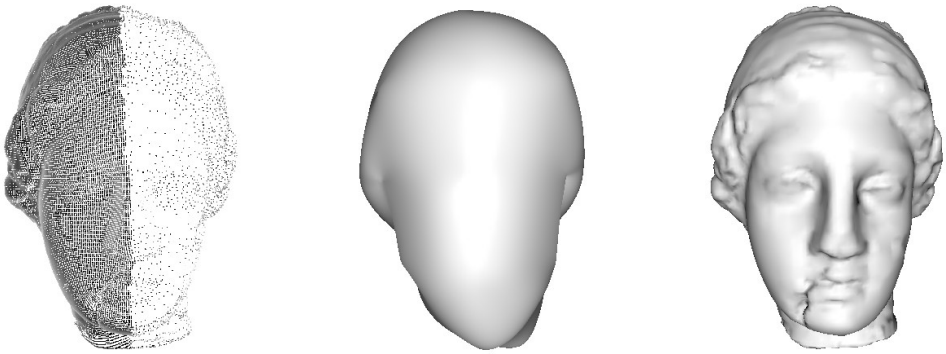
input points reconstructed surface

Figure 5.3: Reconstruction of Knot Object



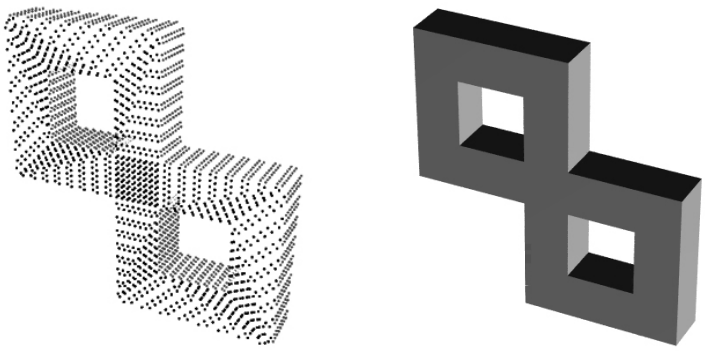
input points reconstructed surface

Figure 5.4: Reconstruction of Bunny Object



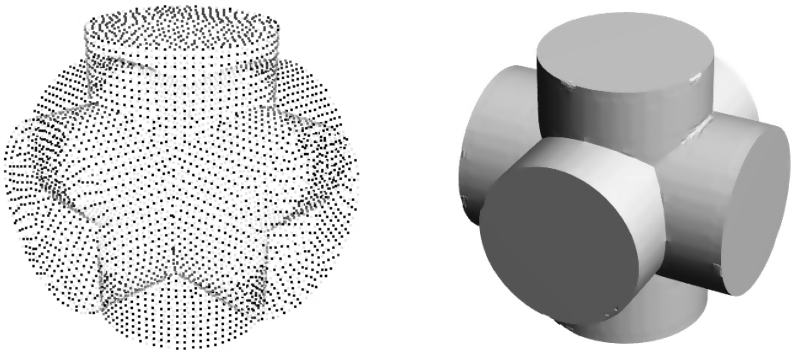
points depth 2 depth 7

Figure 5.5: Reconstruction of Venus Object



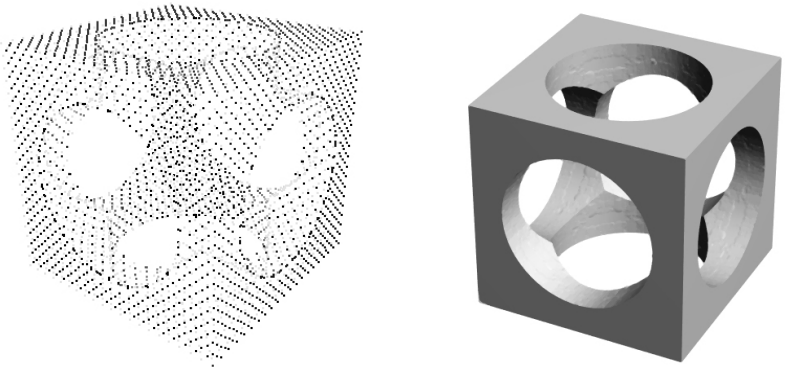
input points implicit surface

Figure 5.6: Reconstruction of Two-torus Object



input points implicit surface

Figure 5.7: Reconstruction of First Boolean Object



input points implicit surface

Figure 5.8: Reconstruction of Second Boolean Object

Appendix A

Implementation

This chapter deals with the specification and implementation of software for implicit surface reconstruction. The software implements the algorithm described in two previous chapters. The results of reconstruction can be found in chapter 5.

On input, we have set of points in 3D space with normals in file .pwn . The output is an implicit function accessed through the function in library. The library and its using is described in following paragraph. The visualization of implicit surface is done using third-party module.

Core Class Design. To describe the internal design we use the Unified Modeling Language (UML) class diagram.

All classes containing an implicit surface are derived from a common base class `ImplicitSurface`. It can compute the value and the gradient in any given point. Local approximation classes are derived from class `Approximation`. They are used to compute approximating function to the set points. The classes reconstructing the objects using local approximations have common class `SurfaceReconstruction`. The implementation of MPUI algorithm is in class `MPUIReconstruction` is derived from class `RBFReconstruction` containing common radial basis function reconstruction. The input points are stored in class `PointSet` and the extending class `PointSetExtension` contains the optimization structure for collecting the points for range queries.

Used Software. The basic platform for the testing the application is MS Windows XP. The library is written in ANSI C++ and compiler with Microsoft C++ compiler.

We use the OpenGL library for the visualization of mesh and we use the Bloomenthal's polygonizer to obtain the mesh. To raytrace the images we use our own plugin for POV-Ray with Isosurface patch.

We use the library of numerical algorithms from Numerical Recipes library.

For writing this thesis text we use the LaTeX in distribution TeX Live 5.

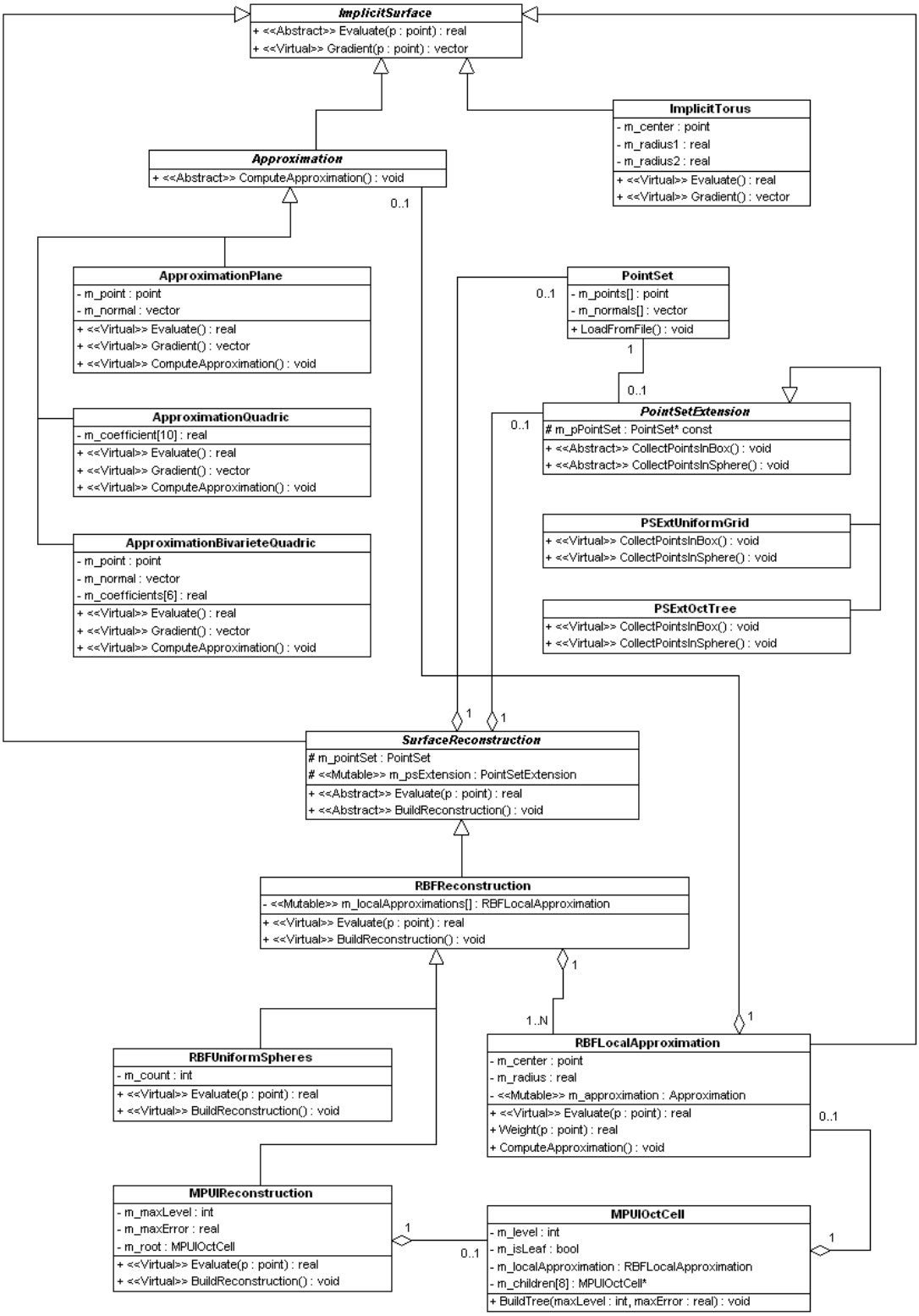


Figure A.1: UML Class Diagram

Appendix B

Compact Disc

The integral part of our work is the included compact disc. Here is the contents.

`Root` Directory.

- This master thesis text in pdf format with color pictures and hyperlinks.
- The poster submited to the SCCG 2005 in Budmerice.
- The paper prepared for the ŠVK 2005, FMFI UK, Bratislava.

Directory `Software`.

- Library for implicit surface reconstruction and the documentation.
- Demonstration application for the reconstruction surface and the tutorial.
- Source code of all software for implicit surface reconstruction library.
- Sample data (point sets with normals) for testing the application.

Directory `Papers`.

- The third-party papers related to the implicit surface reconstruction.
- Saved webpages that were interesting.

Directory `Others`.

- Old mirrors of the webpages of our master thesis.
- Other stuff.

Bibliography

[ABCO+02] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, and C. T. Silva. Computing and rendering point set surfaces. *IEEE TVCG*, 9:3–15, 2002.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of SIGGRAPH*, pages 415–421, 1998.

[AGP+02] M. Alexa, M. Gross, M. Pauly, H. Pfister, M. Stamminger, and M. Zwicker. Point-based computer graphics. In *Eurographics Tutorial Notes*, 2002.

[BG97] R. K. Beatson and L. Greengard. A short course on fast multipole methods. In M. Ainsworth, J. Levesley, W.A. Light, and M. Marletta, editors, *Wavelets, Multilevel Methods and Elliptic PDEs*, pages 1–37. Oxford University Press, 1997.

[Bli82] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[Blo87] J. Bloomenthal. Polygonization of implicit surfaces. Technical report, Xerox Corporation, 1987.

[Blo97] J. Bloomenthal. *Introduction to Implicit Surfaces*. Morgan Kaufmann, 1997.

[BMR+99] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):266–286, 1999.

[BX01] C. L. Bajaj and G. Xu. Regular algebraic curve segments 3: Applications in interactive design and data fitting. *Computer Aided Geometric Design*, 18:149–173, 2001.

[Car98] N. L. Carothers. *A Short Course on Approximation Theory*. Department of Mathematics and Statistics, Bowling Green State University, 1998.

[CBC+01] 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. In *Proceedings of SIGGRAPH*, pages 67–76, 2001.

[CFGN01] P. Chalmovianský, A. Ferko, R. Galbavý, and L. Niepel. *Zložitosť geometrických algoritmov*. Univerzita Komenského Bratislava, 2001.

[Cur97] B. Curless. *New Methods For Surface Reconstruction From Range Images*. Doctor of philosophy, Stanford University, June 1997.

[DGH01] T. K. Dey, J. Giesen, and J. Hudson. Delaunay based shape reconstruction from large data. In *IEEE Symposium on Parralel and Large Data Visualisation*, pages 19–27, 2001.

[EM94] H. Edelsbruner and E. P. Mcke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[Har98] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14:95–108, 1998.

[Has03] M. Hasan. *An Efficient F-rep Visualization Framework*. Master thesis, Faculty of Mathemetics, Physics and Informatics, Comenius University, Bratislava, Slovakia, August 2003.

[HI97] A. Hilton and J. Illingworth. Marching triangles: Delaunay implicit surface triangulation. Technical report, CVSSP, 1997.

[Hop94] H. Hoppe. *Surface Reconstruction from Unorganized Points*. Doctor of philosophy, University of Washington, 1994.

[JF02] B. Juttler and A. Felis. Least-squares fitting of algebraic spline surfaces. *Adv. Comput. Math*, pages 1–20, 2002.

[KBSS] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data.

[Mic] Digital Michelangelo Project, http://graphics.stanford.edu/projects/mich/.

[OBA+03] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. In *Proceedings of SIGGRAPH*, volume 22, pages 463–470, 2003.

[OBS04] Y. Ohtake, A. Belyaev, and H.-P. Seidel. 3D scattered data approximation with adaptive compactly supported radial basis functions. In *Shape Modeling International*, pages 31–39, 2004.

[Oht] Multi-level Partition of Unity Implicits, http://www.mpi-sb.mpg.de/~ohtake/mpu_implicits/.

[PASS95] A. Paško, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[Pau03] M. Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. Doctor of sciences, Federal Institute of Technology (ETH) of Zurich, June 2003.

[PKG03] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Eurographics 2003*, 22(3), 2003.

[POV] POV-Ray, http://www.povray.org.

[Pra87] V. Pratt. Direct least-squares fitting of algebraic surfaces. In *Proceedings of SIGGRAPH - Computer Graphics*, pages 145–152, 1987.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C: the art of scientific computing, Second Edition*. Cambridge University Press, 1992.

[SA01] E. Galin S. Akkouche. Adaptive implicit surface polygonization using marching triangles. *Computer Graphics Forum*, 20(2):67–80, 2001.

[SPOK95] V. Savchenko, A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14:181–188, 1995.

[SVD02] Y. Song, J.S.M. Vergeest, and R. Dumitrescu. Feature recognition for freeform surface design. *4th Workshop on Integrated Product Development*, 2002.

[Tau91] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.

[TO02] G. Turk and J. F. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, October 2002.

[VKKM] G. Varadhan, S. Krishnan, Y.J. Kim, and Dinesh Manocha. Feature-sensitive subdivision and isosurface reconstruction.

[VQ] Vector Quantization, http://www.data-compression.com/vq.shtml.

[XBC00] G. Xu, C. L. Bajaj, and C. I Chu. Regular algebraic curve segments 2: Interpolation and approximation. *Computer Aided Geometric Design*, 17:503–519, 2000.

[XBX00] G. Xu, C. L. Bajaj, and W. Xue. Regular algebraic curve segments 1: Definitions and characteristics. *Computer Aided Geometric Design*, 17:485–501, 2000.

[XMQ] H. Xie, K. T. McDonnell, and H. Qin. Surface reconstruction of noisy and defective data sets.

[Zha97] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *International Journal of Image and Vision Computing*, 15(1):59–76, 1997.