**TEL-AVIV UNIVERSITY**

The Raymond and Beverly Sackler Faculty of Exact Sciences
School of Computer Science

# POINT SET SURFACES

Submitted in partial fulfillment of the requirements
for the degree of “doctor of philosophy”

By
SHACHAR FLEISHMAN

This work was supervised by
Professor DANIEL COHEN-OR

Submitted to the senate of Tel-Aviv University
October, 2003

# Abstract

In this work, we introduce a surface representation that is composed of a set of points in 3D space. The set of points can be the raw output of a 3D scanner or generated during an interactive modeling session. The idea of representing surfaces with 3D points has been suggested twenty years ago, but until recently it has not been widely used. Recent advances in 3D scanning along with the memory capacity and processing speed of today's personal computers rejuvenated point-based object modeling. Most point-based modeling algorithms represent a surface as a union of discs. This object representation requires applying various filtering techniques to achieve high-quality real-time rendering. The focus of this work is on the geometric modeling aspects of representing surfaces with points.

We introduce a new point-based surface representation which we call *point set surface* (PSS) that is globally smooth and at the same time computed locally. This surface representation combines the advantages of point-based modeling with solid mathematical basis with guaranteed geometric properties.

## Point set surfaces

Point set surfaces are defined using an intriguing idea of a projection operator. The projection operator takes a point near the surface and translates it to the surface; the surface is then defined as the set of points that project to themselves. The projection operator is based on the moving least squares (MLS) paradigm, where a function is fitted for each point. Unlike parametric surfaces, where a function is fitted over an area of the surface and continuity is archived by constructing functions with the desired continuity at the boundaries, in the MLS paradigm, a function at a point is approximated using a linear combination of smooth functions. The computation of the projection operator is a two-step procedure: first a local reference domain is fitted to the neighborhood of a point; then a polynomial is fitted over the reference domain. The result is a $C^\infty$ continuous, globally smooth surface. In contrast to radial-basis function interpolation, the computation of the projection operator is local.

The computation of the projection operator involves assigning weights to neighboring points using a smooth weight function. The PSS can either interpolate or approximate the

input set of points. For noisy input or if the user wishes to smooth a surface, we define an approximating weight function. The scale of the weight function controls the amount of smoothing. For clean data, an interpolating weight function is defined. The intrinsic geometric attributes of a PSS can be approximated by the attributes of the locally fitted polynomial.

Finally, we present tools for resampling a PSS. By resmapling a surface to meet the screen resolution and an accurate normal, we obtain a high-quality rendering algorithm.

This part of our work was first presented in *IEEE Visualization 2001* and an extended version appeared in *IEEE Transactions on Computer Graphics and Visualization*.

# Progressive point set surfaces

Progressive point set surfaces (PPSS) is a geometric modeling technique that represents an object in levels, from coarse to fine and from smooth to detailed. We construct a progressive representation of a PSS by first decimating the object to form a simplified and smoothed version of the object. Then, we incrementally refine the base object by inserting new points with additional details. The refinement procedure implicitly adds new points near the surface and projects them both on the base point set and on the input model. A detail vector is then added to the point that was projected on the base point set. The detail vector is equal to the difference between the two projections of the point.

We decompose the detail of an inserted point to a tangential and normal components. The tangential component is implicitly defined by a set of refinement rules and the normal component is the only part that needs to be encoded. We use an approximated projection procedure that projects a point on the input model using the reference domain of the current coarse object. This allows us to store the normal component using a single scalar value, leading to a highly compact object representation.

This part of our work was published in *ACM Transactions on Graphics*.

# Bilateral mesh denoising

In this chapter, we introduce a feature preserving surface denoising algorithm. Unlike denoising a surface using PSS, this algorithm preserves sharp features. Robust statistics algorithms ignores outliers in the sampled data; the idea behind this algorithm is that two vertices on two opposing sides of an edge treat one another as outliers. We adapt an image denoising algorithm (the bilateral filter) that is related to robust statistics to surfaces. The adaptation is inspired by the computation of the PSS projection operator. For each vertex

we build a local reference domain and project the neighborhood of the vertex onto the reference domain; the use of a local frame decomposes the local neighborhood to a tangential and normal components, where the tangential component represents the parameter information and the normal component represents the geometric information. We then apply the bilateral filter in this local frame to the geometric component.

In addition, we show how to select the parameters of the algorithm using an interactive technique. The user marks a part of the model that is supposed to be smooth by selecting a point and a radius for applying the smoothing. The remaining parameters are inferred from the neighborhood of the selected point.

This part was presented in *SIGGRAPH 2003*, special issue of *ACM Transactions on Graphics*.

# Contents

**Abstract** **ii**

**1 Overview** **1**
1.1 The model acquisition pipeline . . . . . 1
1.2 Surface representations . . . . . 2
1.2.1 Parametric surface representations . . . . . 3
1.2.2 Implicit surface representations . . . . . 4
1.2.3 Point-based surface representation . . . . . 6
1.3 Summary . . . . . 8

**2 Point Set Surfaces** **9**
2.1 Related work . . . . . 12
2.1.1 Consolidation . . . . . 12
2.1.2 Point sample rendering . . . . . 14
2.2 Defining the surface - point projection . . . . . 15
2.2.1 The projection procedure . . . . . 15
2.2.2 Properties of the projection procedure . . . . . 16
2.2.3 Computing the projection . . . . . 17
2.2.4 Data structures and trade-offs . . . . . 20
2.3 Approximation error . . . . . 21
2.4 Generating the representation point set . . . . . 22
2.4.1 Down-sampling . . . . . 22
2.4.2 Up-sampling . . . . . 24
2.5 Rendering . . . . . 25
2.5.1 Culling and view dependency . . . . . 25
2.5.2 Sampling additional points . . . . . 26
2.5.3 Grid pyramids . . . . . 27
2.5.4 Results . . . . . 27

**3 Progressive Point Set Surfaces** **32**
3.1 Related Work . . . 34
3.2 MLS surfaces . . . 36
3.3 Progressive point set surfaces . . . 36
3.3.1 The refinement operator . . . 37
3.3.2 Constructing the base point set . . . 39
3.4 The PPSS encoding . . . 40
3.5 Results . . . 41
3.6 Discussion . . . 45

**4 Bilateral Mesh Denoising** **49**
4.1 Previous work . . . 51
4.2 Bilateral mesh denoising . . . 51
4.2.1 Bilateral filtering of images. . . . 52
4.2.2 Algorithm . . . 52
4.2.3 Mesh shrinkage and vertex-drift . . . 53
4.2.4 Handling boundaries . . . 54
4.2.5 Parameters . . . 55
4.3 Results . . . 57
4.4 Summary . . . 58

**5 Conclusions and future work** **59**
5.1 Point set surfaces . . . 59
5.2 Progressive point set surfaces . . . 60
5.3 Bilateral mesh denoising . . . 61

**Hebrew Abstract** **ב**

# List of Figures


1.1 The model acquisition pipeline . . . . . 2

2.1 Sample point set object . . . . . 10
2.2 An illustration of the paradigm . . . . . 11
2.3 The MLS projection procedure . . . . . 15
2.4 The effect of the parameter $h$ . . . . . 18
2.5 Uneven sampling . . . . . 23
2.6 Point set object decimation . . . . . 23
2.7 Denoising concept . . . . . 29
2.8 The up-sampling process . . . . . 29
2.9 Upsampling to screen-space resolution . . . . . 30
2.10 The patch size of a polynomial . . . . . 30
2.11 Comparison of rendering using splatting, polygons and PSS . . . . . 31
2.12 Rendering of polygons and point sets with environment maps . . . . . 31

3.1 The progressive series of the Isis model . . . . . 33
3.2 An illustration of the point set refinement process . . . . . 38
3.3 Close-ups of smooth renderings . . . . . 39
3.4 Color-coding of the magnitude of the displacement . . . . . 40
3.5 Rate distortion curve . . . . . 41
3.6 Comparison of meshes using Metro . . . . . 44
3.7 A progressive series of the Venus and Dragon models . . . . . 45
3.8 Rendering of intermediate representation of the Venus model . . . . . 46
3.9 Reconstruction of sharp edges and boundaries . . . . . 47

4.1 Denoising a scanned model . . . . . 50
4.2 The shrinkage problem . . . . . 54
4.3 Comparison with AFP . . . . . 55
4.4 Results of denoising the face model . . . . . 56
4.5 Denoising of CAD-like model . . . . . 56
4.6 Results of denoising the Fandisk . . . . . 57

5.1 Denoising with discontinuities in normals . . . . . 61

# Chapter 1

# Overview

Representing objects as a set of points in three dimensions received increasing interest in recent years for two main reasons: (i) point-based surface representation allows editing of objects without the necessary bookkeeping of topological information inherent in traditional mesh representations. (ii) highly detailed objects are represented by very dense sampling of the surface, making the connectivity information redundant. The geometric model that is used by most point-based algorithms is a union of discs. This model is useful for rendering and interactive editing purposes; but, since a union of discs is not a continuous or smooth geometric model, rendering requires filtering of points, and geometric processing algorithms are not applied directly to point-based models. In this work, we introduce a point-based geometric representation that is locally defined and globally smooth.

One source for point-based objects are three-dimensional scanners that sample the surface of an object to generate a set of points in space. This set of points is termed *scattered data*, *unorganized points*, or a *point sets*, we will use these terms interchangeably. Once an object has been scanned, it goes through a number of transformations to form another representation that is suitable for further processing and rendering. A natural representation for such on object is simply the set of point that were obtained from the scanner, which is the goal of this work.

## 1.1 The model acquisition pipeline

The process of producing a mathematical representation from a scanned object, *the model acquisition pipeline*, consists of several stages (see Figure 1.1). First, an object is scanned from several viewing directions to cover the object. Then the different scans are registered, transforming the points from all scans to a single coordinate system. Next, the noise that is introduced by the scanning process is removed, and finally, the cloud of points is converted to a some surface representation. In this work, we are mostly interested in the

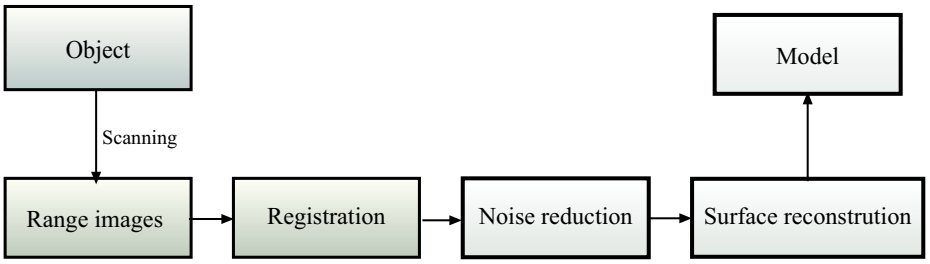



Figure 1.1: The model acquisition pipeline.

surface representation, its usability for different applications, and its resilience to noise. For more information on three dimensional registration, see [Rusinkiewicz 2001] and references therein.

Resilience to noise is achieved by smoothly approximating the set of points. Interpolating methods preserve the geometry of the input set of points and thus require another denoising step. The model acquisition pipeline as described here is a conceptual paradigm; naturally, different surface representations and reconstruction algorithms omit different stages or perform them in different order.

## 1.2 Surface representations

Some operations are more natural to one surface representation than to another; for example it is more natural to perform constructive solid geometry (CSG) operations such as union and intersection of objects in implicit surface representation than in a triangular mesh representation. In this section we give a short overview of three types of surface representations: (i) parametric surfaces, where we concentrate on polygonal meshes that are the most common representation for scanned object reconstruction. (ii) implicit function representation, where we concentrate on radial basis function interpolation (RBF) for which modern algorithms were developed for fast reconstruction and evaluation. (iii) point-based object representation. In the context of surface reconstruction and representation, there is a close relationship between polygonal mesh representation and implicit surface representation. The first is used as a representation for real-time rendering for the second, while the second is used as a first stage in reconstructing polygonal meshes from unorganized points.

For each surface representation, we briefly review a number of representative reconstruction algorithms from scattered data, the ways surface representation and the reconstruction algorithms cope with noise, the smoothness of the reconstructed surface, and methods for representing derivative discontinuities, i.e. edges, creases and corners. The final point we examine is the computational complexity of performing operations on the surface and in particular rendering the surface. For more in depth discussion of scattered

data techniques, see [Lodha and Franke 1999].

### 1.2.1 Parametric surface representations

A parametric surface representation defines an object by modeling its boundary. It is defined by a function over a parameter domain $F(u, v) \mapsto \Re^3$. An object, defined by its boundary, cannot always be represented using a single planar parameter domain and therefore an object's manifold is represented as the union of parametric surfaces named *patches*. Every two patches should overlap only at their boundaries and never intersect each other. A surface reconstruction algorithm, must make sure that the reconstructed surface is continuous, and it is further desirable that the surface is $C^1$ or $C^2$ smooth.

#### Polygonal meshes

A special case of parametric surface representation is a polygonal mesh, typically a mesh of triangles. Polygonal meshes are represented by the pair $\langle V, F \rangle$, where $V$ is the set of vertices of the triangle mesh, representing the geometry of the object, and $F$ is the set of faces of the mesh, that is, a union of ordered lists (triplets in triangle meshes) of vertices. The edges of the faces represent the topology of the mesh.

Polygonal meshes are simple to represent. They are rendered using specialized hardware, and using subdivision schemes, they can represent smooth surfaces [Catmull and Clark 1978; Warren and Weimer 2001].

Converting scattered data to polygonal surfaces is often performed by first defining an implicit function, either using the input set of points [Hoppe et al. 1992] or using a volumetric representation [Curless and Levoy 1996]. The function is sampled and converted to a mesh using the *marching cubes* algorithm [Lorensen and Cline 1987]. Hoppe et al. [1994] use the above recovered polygonal mesh as an initial estimate of the surface which is further refined by a nonlinear optimization algorithm. They optimize the following constraints: smoothness using subdivision rules, preservation of sharp features, and simplicity of the object. Curless and Levoy [1996] form a volumetric representation of the signed distance function. They interpolate each new scan into the already computed signed distance function with weights that are based on the confidence of the sample in order to create a consensus surface.

Another approach for creating a mesh from scanned data was taken by Turk and Levoy [1994]. Each scan, which is a depth image, is meshed. The meshes from different scans are merged the together in a process they name *mesh zippering*.

The question of sampling criteria was considered by Amenta and Bern [1999]. They concentrate on a surface reconstruction algorithm that given a "good sample" produces "correct reconstruction". Amenta and Bern provide both theoretical guarantees, assuming

noise free samples [Amenta and Bern 1999], and practical experience of using their algorithm [Amenta et al. 1998]. They define a good sample as one in which the sampling density is proportional to the distance to the medial axis of the object. A good reconstruction is defined as one that is topologically equivalent to the input model.

Recently Igarashi and Hughes [2003] defined a smooth surface from a mesh that was used in a free-form modeling system for mesh fairing. They locally fit a quadric to the vertices of a mesh. A point on the mesh is then evaluated as a linear combination of a projection of the point on the quadrics associated with the vertices that are closest to the point.

#### High order parametric surfaces

High order parametric surfaces are defined by polynomial approximation or interpolation using a set of control points. The most widely used are tensor product of smooth curves such as B-splines or NURBS. Patches are connected together, maintaining continuity of derivatives.

Since the pioneering work of Pierre E. Bézier [1968], surface patches became common modelling tools for computer aided design and manufacturing (CADM), and for modelling of smooth geometric objects. Their major advantage is that they are intuitive to edit using a small number of control-points. Surfaces that are acquired by 3D scanners are highly detailed and therefore difficult to edit [Greiner et al. 1996]. For this reason, they are not widely used for detailed geometric surface representation and reconstruction. Krishnamurthy and Levoy [1996] reconstruct a scanned model using a hybrid of surface patches and geometric details. Each patch is a smooth approximation of the underlying surface that is further refined using detail vectors that are stored in an image format.

### 1.2.2 Implicit surface representations

Implicit surfaces are defined as the zero set of a function $F(\mathbf{x}) : \Re^3 \mapsto \Re$. They can be modeled using basic geometric primitives such as spheres, Gaussian blobs etc. Computing CSG operations such as union, intersection and subtraction of implicit surfaces is simple, enabling the construction of complex objects. Rendering implicit surfaces is not a real-time process, though they can be sketched using particles [Witkin and Heckbert 1994].

Reconstructing a surface from 3D scanned data $S = \{p_i\}$ of size $N = |S|$ can be viewed as a scattered data interpolation problem. An implicit function solution to scattered data interpolation uses radial basis functions (RBF). A radial basis function is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{N} a_i \cdot \phi(\|\mathbf{x} - \mathbf{p_i}\|) + \sum_{i=1}^{M} b_i \cdot P(\mathbf{x}), \tag{1.1}$$

where P is a polynomial of degree $m$ and $M = \frac{m(m+1)}{2}$. In order to define an RBF interpolation to an input model, one has to solve a linear system of $N + M$ equations. In order to avoid the trivial solution, a number of points that are inside and outside the model with negative and positive values respectively are added. Altogether, one has to solve a linear system of equations with up to $3N + M$ variables.

The above scattered data interpolation problem has an infinite number of solutions. A common solution is to select a radial function that minimizes the total curvature, known as thin-plate spline interpolation, whose definition in two dimensions is:

$$\phi(x) = x^2 \log(x), \tag{1.2}$$

and in three dimensions:

$$\phi(x) = x. \tag{1.3}$$

Notice that these functions have a support that grows as a point gets farther from the center, forming a full symmetric positive definite matrix.

Solving Eq. 1.1 is an $O(n^3)$ problem that can be solved for objects of moderate size [Turk and O'brien 2002]. A number of algorithms for fast solution and evaluation of radial basis functions were introduced in recent years. Carr et al. [2001] applied fast evaluation of radial basis functions [Beatson and Newsam 1992] with $O(N \log(N))$ time for solving Eq. 1.1 and $O(1)$ with setup time of $O(N \log(N))$ for evaluating the function. The main idea in their solution is to hierarchically cluster points that are far from a point to a single representative. In addition, they developed a method for approximating a set of points using RBF and a single parameter that controls the smoothness of the surface.

Compactly supported radial basis functions is a different approach for the computational and stability problems of direct solutions for Eq. 1.1 [Morse et al. 2001]. Compactly supported RBFs form a sparse matrix for Eq 1.1 by using a radial basis function with finite or truncated support. This sparse matrix can be solved in $O(N^{1.5})$, and evaluating the function becomes an $O(\log(N))$ problem. The cost of using compactly supported RBFs is an additional parameter that is the size of the support.

Radial basis function interpolation forms a smooth function, which is desirable in many cases. However some objects have derivative discontinuities, i.e. edges, creases and corners. Dinh et al. [2001] used anisotropic basis functions for surface interpolation. They scale the distance from a center anisotropically based on an eigenvector analysis of the neighborhood of the point.

Recently, Ohtake et al. [2003] introduced an implicit function representation from points that requires only local computation. Starting with a bounding box of the input set of points, a function is fitted to the points. If the error between the function and the points is larger than some threshold, the box is subdivided into eight sub-boxes and the process is repeated recursively. Three types of functions are fitted: a parametric polynomial, a quadric similar to Igarashi and Hughes [2003], or an intersection of two functions

that is used to captures sharp features. The implicit model is then defined using partition of unity of the functions in the neighborhood of a point.

Performing constructive solid geometry (CSG) operations on implicit surfaces is simple using *min*, *max*, and *negate* operators. Filleting and blending operations are also easily performed [Bloomenthal et al. 1997]. With the exception of the work of Ohtake et al. [2003], any modification to the points must be followed by solving a matrix, an operation that is too costly for an interactive editing session.

Implicit functions are rendered by ray-tracing [Hart 1996], or by first converting them to a polygonal mesh [Bloomenthal 1994; Bloomenthal 1988], and then rendering them using standard graphics hardware.

### 1.2.3 Point-based surface representation

Research on point-based object representation was mainly concerned with fast rendering of complex objects or scenes. The idea of using points as graphic primitives was first considered by Levoy and Whitted [1985]. Chen and Williams [1993] introduced a walk-through system that used images with depth as a model. Images with depth representation (commonly known as *image based rendering* (IBR)) are mostly concerned with reusing real or synthetic images for fast rendering, exploiting the coherence inherent in images. Researches in image based rendering are mostly interested in capturing objects and environments [Gortler et al. 1996; McMillan and Bishop 1995], data structures for representing such models [Shade et al. 1998], good sampling [Stuerzlinger 1999; Fleishman et al. 2000], and surface sampling algorithms that lead to rendering without introducing holes or cracks in the image [Chai et al. 2000].

#### Rendering

Grossman and Dally [1998] suggested to represent complex objects using points for real-time rendering with no specialized hardware. They presented an object sampling criteria such that the rendered image contains no holes. The main idea in this work is to cover the output image with an virtual triangle mesh such that the length of triangle edges are less than the side size of a pixel.

Rusinkiewicz and Levoy [2000] introduced a point-based system for interactive rendering and compact encoding of objects. They model an object as a union of overlapping spheres. A hierarchy of spheres that utilizes relative coordinates for compact encoding is built from the union of spheres in a bottom-up manner. The model is rendered by traversing the hierarchy from the root downwards, stopping whenever a leaf node is reached, the projected size of a node is less than a pixel or the time budget is reached.

Pfister et al. [2000] model an object as a union of discs. They introduce an image space

elliptical weighted average (EWA) filter for high-quality point-based rendring. In addition, they use an octree for point-based level of detail rendering and fast culling.

Kalaiah and Vershney [2003] store at every point the principal directions and curvatures in addition to color and normal information. Areas of low curvature should have low sampling density, while areas of high curvature should be sampled more densely; therefore, they simplify an object by discarding points whose geometric information is represented by their neighbors within a user specified error. For hardware rendering, they precompute normal maps for different curvature values.

Schaufler and Wann-Jensen [2000] introduced a ray tracing algorithm for point-sampled geometry. They also represent an object as a union of discs, implemented by tracing a cylinder with some predefined radius until a point is hit. Their visual results are very appealing, however their model may change when the camera parameters change. This problem can be solved by filtering or by using a different geometric representation.

### Modeling and editing

Adams and Dutre [2003] introduced interactive boolean operations on point-based models. They paint an octree that bounds the object with three colors, one for cells that are completely outside the object, one for cells that completely inside the object and one for cells that contain part of the boundary of the object. For CSG operations, the octrees of the two objects are traversed, quickly handling cells that are completely inside or outside of the objects. The remaining cells are handled by defining additional clipping planes based on the points in the cell. In a second step, for better approximation of the sharp curve that is formed by the CGS operations, they resmaple the surface near the intersection curve by decreasing the radius of points and adding new points with smaller radius.

Pointshop3D [Zwicker et al. 2002] is a point-based surface editing system that supports surface painting and carving. The system is based on parameterization and resampling operators. To apply a tool to the surface, both the tool and the surface patch are parameterized over the same domain, then they are resampled to the same resolution and finally, the tool is applied to the patch in the parametric domain. Pauly et al. [2003] extended that system with boolean operations and free-form deformations with point set surfaces [Alexa et al. 2003] as the underlying surface representation. Boolean operations are performed by classifying points as inside/outside of the object. The classification is accelerated by exploiting local coherence; every point that is contained in a sphere with a radius that is equal to the distance between a classified point and the closest point on the surface has the same classification as the point. They also recover sharp features by computing and adding new points along the intersection curve. In a final step, they upsample or downsample [Pauly et al. 2002] the object to maintain the desired sampling density. The inside/outside classification is also used for topology control in interactive free-form deformation of an object.

## 1.3 Summary

In this work, we introduce a geometric surface definition from point sets which we call *point set surfaces* (PSS). The surface is defined using a projection operator that utilizes the moving least squares (MLS) paradigm with local parameter space [Levin 2003]. Given a point near the surface, the projection operator translates the point to the surface, and the surface is defined as the set of points that project to themselves. The computation of the projection operator is a two step procedure: first a local reference domain is fitted to the neighborhood of a point. Then a polynomial is fitted over the reference domain. The result is a $C^\infty$ continuous, globally smooth surface. The computation of the projection operator is local, so that moving a point during an editing session, for example, does not affect the entire surface. The projection operator relies on a single parameter that is the average of point spacing. The smoothness of the surface is controlled by this parameter, allowing reconstruction of objects from clean, as well as noisy point sets.

In Chapter 2, we describe the point set surface, show how to compute the projection operator, use our surface definition for smoothing and denoising surfaces, and provide tools for resampling the surface. Surface resampling to screen resolution gives rise to a high quality point-based rendering algorithm.

In Chapter 3, a progressive representation of point set surfaces is introduced. Progressive point set surfaces (PPSS) are a multilevel point-based surface representation. A PPSS representation of an object is constructed by creating a base point set that is a decimated and a smooth version of the input model. Next, using a refinement operator, we insert new detail points that are a combination of a new point that is projected on the surface with an additional detail vector.

We decompose the detail of an inserted point into tangential and normal components. The tangential component is implicitly defined by the refinement operator and the normal component is the only part that needs to be encoded. We use an approximated projection procedure that projects a point on the input model using the reference domain of the coarser object. This allows us to store the normal component using a single scalar value, leading to a highly compact object representation.

The PSS is a smooth surface representation, thus it smoothes sharp features when used for denoising. In Chapter 4, we introduce a feature-preserving surface denoising algorithm. We use the idea of local maps from the projection operator and the separation to tangential and normal components from the PPSS definition in order to apply an image denoising algorithm locally to every point in the input model. In particular, we adapt the bilateral filter [Smith and Brady 1997; Tomasi and Manduchi 1998] image deoising algorithm to surfaces. The result is a very simple and fast algorithm for surface denoising. We sum up with a discussion and suggest ideas for future work in Chater 5.

# Chapter 2

# Point Set Surfaces

Point sets are receiving a growing amount of attention as a representation of models in computer graphics. One reason for this is the emergence of affordable and accurate scanning devices generating a dense point set, which is an initial representation of the physical model [Levoy et al. 2000]. Another reason is that highly detailed surfaces require a large number of small primitives, which contribute to less than a pixel when displayed, so that points become an effective display primitive [Pfister et al. 2000; Rusinkiewicz and Levoy 2000].

A point-based representation should be as small as possible while conveying the shape, in the sense that the point set is neither noisy nor redundant. It is important to have tools which adequately adjust the density of points so that a smooth surface can be well-reconstructed. Figure 2.1 shows a point set with varying density. Amenta et al. [1998] have investigated the problem from a topological point of view, that is, the number of points needed to guarantee a topologically equivalent reconstruction of a smooth surface. Our approach is motivated by differential geometry and aims at minimizing the geometric error of the approximation. This is done by locally approximating the surface with polynomials using moving least squares (MLS).

We understand the generation of points on the surface of a shape as a sampling process. The number of points is adjusted by either up-sampling or down-sampling the representation. Given a data set of points $P = \{\mathbf{p}_i\}$ (possibly acquired by a 3D scanning device), we define a smooth surface $S_P$ (***MLS surface***) based on the input points (the definition of the surface is given in Section 2.2). We suggest replacing the points $P$ defining $S_P$ with a reduced set $R = \{\mathbf{r}_i\}$ defining an MLS surface $S_R$ which approximates $S_P$. This general paradigm is illustrated in 2D in Figure 2.2: Points $P$, depicted in purple, define a curve $S_P$ (also in purple). $S_P$ is resampled with points $\mathbf{r}_i \in S_P$ (red points). This typically lighter point set called the *representation* points now defines the red curve $S_R$ which approximates $S_P$.

Figure 2.1: A point set representing a statue of an angel. The density of points and, thus, the accuracy of the shape representation are changing (intentionally) along the vertical direction.

The technique that defines and resamples $S_P$ provides the following important properties:

**Smooth manifold:** The surface defined by the point set is guaranteed to be a 2-manifold and $C^\infty$ smooth, given that the points are sufficiently close to the surface being represented.

**Bounded sampling error:** Let $S_R$ be defined by the set of representation points $\mathbf{r}_i \subset S_P$. The representation has bounded error $\epsilon$, if $d(S_P, S_R) < \epsilon$, where $d(\cdot, \cdot)$ is the Hausdorff distance.

**Local computation:** For computing a point on the surface only a local neighborhood of that point is required. This results in a small memory footprint, which depends only on the anticipated feature size and not the number of points (in contrast to several other implicit surface definitions, e.g. those based on radial basis functions).

In the context of surface consolidation (see Section 2.1), our approach has two important features:

**Noise reduction:** Filtering imperfect data and generating a thin point set, in the sense that the points do not deviate from the surface being represented.

**Redundancy reduction:** The point set is down-sampled by removing redundant information introduced by oversampling the surface.

Finally, we present a rendering method that approximates $S_R$ from the local polynomial approximations offering. The bounded sampling error substantiates our claimed for the following properties of our rendering scheme:

**High quality:** Since $S_R$ is a smooth surface, proper resampling leads to smooth silhouettes and normals resulting in superior rendering quality at interactive frame rates.

**Single step procedure:** Resampling respects screen space resolution and guarantees sufficient sampling, i.e. no holes have to be filled in a postprocessing step.

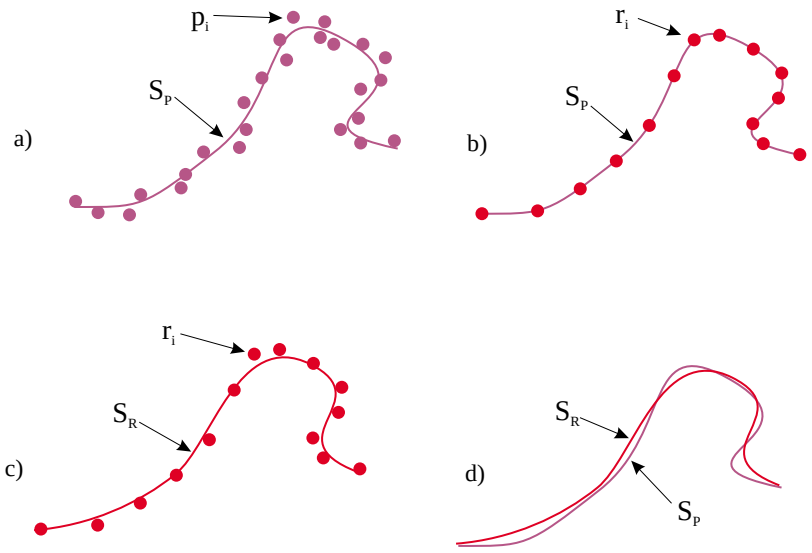



Figure 2.2: An illustration of the paradigm: The possibly noisy or redundant point set (purple points) defines a manifold (purple curve). This manifold is sampled with (red) representation points. The representation points define a different manifold (red curve). The spacing of representation points depends on the desired accuracy of the approximation.

## 2.1 Related work

### 2.1.1 Consolidation

Recent technological and algorithmic advances have improved the process of automatic acquisition of 3D models. Acquiring the geometry of an object starts with data acquisition, usually performed with a range scanner. This raw data contains errors (e.g., line-of-sight error [Hebert et al. 1993; Rutishauser et al. 1994]) mainly due to noise intrinsic to the sensor used and its interaction with the real-world object being acquired. For a non-trivial object, it is necessary to perform multiple scans, each in its own coordinate system, and to register the scans [Besl and McKay 1992]. In general, areas of the objects are likely to be covered by several samples from scans performed from different positions. One can think of the output of the registration as a *thick* point set.

A common approach is to generate a triangulated surface model over the thick point set. There are several efficient triangulation techniques, such as [Amenta et al. 1998; Bajaj et al. 1995; Bernardini et al. 1999; Boissonnat 1984; Gopi et al. 2000]. One of the shortcomings of this approach is that the triangulated model is likely to be rough, containing bumps and other kinds of undesirable features, such as holes and tunnels, and be non-manifold. Further processing of the triangulated models, such as smoothing [Taubin 1995; Desbrun et al. 1999] or manifold conversion [Gueziec et al. 1998], becomes necessary. The prominent difficulty is that the point set might not actually interpolate a smooth surface. We call *consolidation* the process of "massaging" the point set into a surface. Some techniques, such as Hoppe et al. [1992], Curless and Levoy [1996], and Wheeler et al. [1998] consolidate their sampled data by using an implicit representation based on a distance function defined on a volumetric grid. In [Hoppe et al. 1992], the distances are taken as the signed distance to a locally defined tangent plan. This technique needs further processing [Hoppe et al. 1993; Hoppe et al. 1994] to generate a smooth surface. Curless and Levoy [1996] use the structure of the range scans and essentially scan convert each range surface into the volume, properly weighting the multiply scanned areas. Their technique is robust to noise and is able to take relative confidence of the samples into account. The work of Wheeler et al. [1998] computes the signed distance to a consensus surface defined by weighted averaging of the different scans. One of the nice properties of the volumetric approach is that it is possible to prove under certain conditions that the output is a least-square fit of the input points (see [Curless and Levoy 1996]).

The volumetric sign-distance techniques described above are related to a new field in computer graphics called *Volume Graphics* pioneered by Kaufman and colleagues [1993; 1995; 1999] which aims to accurately define how to deal with volumetric data directly, and answer questions related to the proper way to convert between surface and volume representations.

It is also possible to consolidate the point set by performing weighted averaging directly on the data points. In [Turk and Levoy 1994], model triangulation is performed first, then averaging is performed in areas which overlap. In [Soucy and Laurendeau 1992], the data points are first averaged, weighted by a confidence in each measurement, and then triangulated.

Another approach to define surfaces from the data points is to perform some type of surface fitting [Goshtasby and O'Neill 1993], such as fitting a polynomial [Lei et al. 1996] or an algebraic surface [Pratt 1987] to the data. In general, it is necessary to know the intrinsic topology of the data and (sometimes) have a parametrization before surface fitting can be applied. Since this is a non trivial task, Krishnamurthy and Levoy [1996] have proposed a semi-automatic technique for fitting smooth surfaces to dense polygon meshes created by Curless and Levoy [1996]. Another form of surface fitting algorithms couples some form of high-level model recognition with a fitting process [Ramamoorthi and Arvo 1999].

The process of sampling (or resampling) surfaces has been studied in different settings. For instance, surface simplification algorithms [Cignoni et al. 1998] sample surfaces in different ways to optimize rendering performance. Related to our work are algorithms which use particle systems for sampling surfaces. Turk [1992] proposes a technique for computing level of details of triangular surfaces by first randomly spreading points on a triangular surface, then optimizing their positions by letting each point repel their neighbors. He uses an approximation of surface curvature to weight the number of points which should be placed in a given area of the surface. A related approach is to use physically-based particle systems to sample an implicit surface [Witkin and Heckbert 1994; de Figueiredo et al. 1992]. Crossno and Angel [1997] describe a system for sampling isosurfaces, where they use the curvature to automatically modulate the repulsive forces.

Lee [2000] uses a moving-least squares approach to the reconstruction of curves from unorganized and noisy points. He proposes a solution for reconstructing two and three-dimensional curves by thinning the point sets. Although his approach resembles the one used here (and based on theory developed in [Levin 2003]), his *projection* procedure is different, and requires several iterations to converge to a clean point set (i.e., it is not actually a projection, but more of a converging smoothing step).

An alternative to our point-based modelling mechanism is proposed by Linsen [2001]. His work is based on extending the $k$ neighborhood of a point to a "fan" by using an angle criterion (i.e., a neighborhood should cover the full $360$ degrees around). Using this simple scheme, Linsen proposes a variety of operations for point clouds, including rendering, smoothing, and some modelling operations.

### 2.1.2 Point sample rendering

Following the pioneering work of Levoy and Whitted [1985], several researchers have recently proposed using "points" as the basic rendering primitive, instead of traditional rendering primitives, such as triangulated models. One of the main reasons for this trend is that in complex models the triangle size is decreasing to pixel resolution. This is particularly true for real-world objects acquired as "textured" point clouds [McAllister et al. 1999].

Grossman and Dally [1998] presented techniques for converting geometric models into point-sampled data sets, and algorithms for efficiently rendering the point sets. Their technique addresses several fundamental issues, including the sampling rate of conversion from triangles to points, and several rendering issues, including handling "gaps" in the rendered images and efficient visibility data structures. The Surfels technique of Pfister et al. [2000] builds and improves on this earlier work. They present alternative techniques for the sampling of the triangle mesh, including visibility testing, texture filtering, and shading.

Rusinkiewicz and Levoy [2000] introduce a technique which uses a hierarchy of spheres of different radii to model a high-resolution model. Their technique uses small spheres to model the vertices at the highest resolution, and a set of bounding spheres to model intermediate levels. Together with each spherical sample, they also save other associated data, such as normals. Their system is capable of time-critical rendering, as it adapts the depth of tree traversal to the available time for rendering a given frame.

Kalaiah and Varshney [2001] introduced an effective method for rendering point primitives that requires the computation of the principal curvatures and a local coordinate frame for each point. Their approach renders a surface as a collection of local neighborhoods, and it is similar to our rendering technique proposed later, although they do not use dynamic level of detail in their system.

All the above techniques account for local illumination. Schaufler and Jensen [2000] propose to compute global illumination effects directly on point-sampled geometry by a ray tracing technique. The actual intersection point is computed, based on a local approximation of the surface, assuming a uniform sampling of the surface.

Point-based rendering suffers from the limited resolution of the fixed number of sample points representing the model. At some distance, the screen space resolution is high relative to the point samples, which causes undersampling. Tackling this problem by interpolating the surface points in image space is limited. A better approach is to resample the surface during rendering at the desired resolution in object-space, guaranteeing that sampling density is sufficient with respect to the image space resolution.

Hybrid polygon-point approaches have been proposed. Cohen et al. [2001] introduces a simplification technique which transitions triangles into (possibly multiple) points for faster rendering. Their system uses an extension of DeFloriani et al's Multi-Triangulation data structure [Floriani et al. 1997; Floriani et al. 1998]. A similar system has been developed

by Chen and Nguyen [2001] as an extension of QSplat [Rusinkiewicz and Levoy 2000].

## 2.2 Defining the surface - point projection

Our approach relies on the idea that the given point set implicitly defines a surface. We build upon recent work by Levin [2003]. The main idea is the definition of a projection procedure, which projects any point near the point set onto the surface. Then, the MLS surface is defined as the points projecting onto themselves. In the following, we explain the projection procedure, prove the projection and manifold properties, motivate the smoothness conjecture, and, give details on how to efficiently compute the projection.

### 2.2.1 The projection procedure

Let points $\mathbf{p}_i \in \Re^3, i \in \{1, \ldots, N\}$, be sampled from a surface $S$ (possibly with a measurement noise). The goal is to project a point $\mathbf{r} \in \Re^3$ near $S$ onto a two-dimensional surface $S_P$ that approximates the $\mathbf{p}_i$'s. The following procedure is motivated by differential geometry, namely that the surface can be locally approximated by a function.

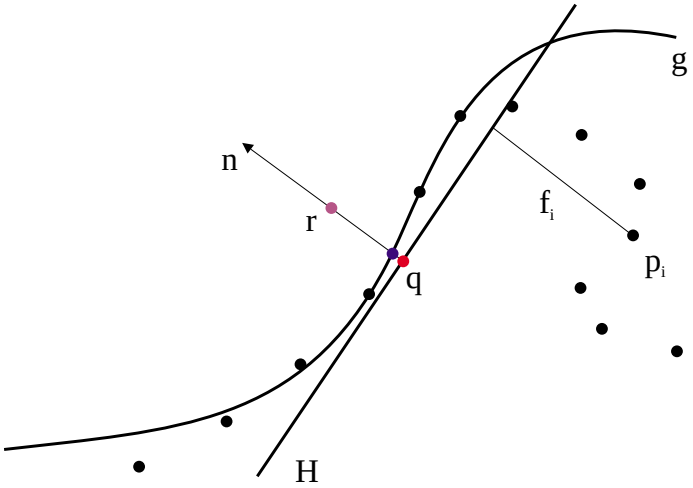



Figure 2.3: The MLS projection procedure: First, a local reference domain $H$ for the purple point $\mathbf{r}$ is generated. The projection of $\mathbf{r}$ onto $H$ defines its origin $\mathbf{q}$ (the red point). Then, a local polynomial approximation $\mathbf{g}$ to the heights $f_i$ of points $\mathbf{p}_i$ over $H$ is computed. In both cases, the weight for each of the $\mathbf{p}_i$ is a function of the distance to $\mathbf{q}$ (the red point). The projection of $\mathbf{r}$ onto $g$ (the blue point) is the result of the MLS projection procedure.

1. **Reference domain:** Find a local reference domain (plane) for $\mathbf{r}$ (see Figure 2.3). The local plane $H = \{\mathbf{x} | \langle \mathbf{n}, \mathbf{x} \rangle - D = 0, \mathbf{x} \in \Re^3\}, \mathbf{n} \in \Re^3, \|\mathbf{n}\| = 1$ is computed

so as to minimize a local weighted sum of square distances of the points $\mathbf{p}_i$ to the plane. The weights attached to $\mathbf{p}_i$ are defined as the function of the distance of $\mathbf{p}_i$ to the projection of $\mathbf{r}$ on the plane $H$, rather than the distance to $\mathbf{r}$. Assume $\mathbf{q}$ is the projection of $\mathbf{r}$ onto $H$, then $H$ is found by minimizing

$$\sum_{i=1}^{N} \left(\langle \mathbf{n}, \mathbf{p}_i \rangle - D\right)^2 \theta\left(\|\mathbf{p}_i - q\|\right) \tag{2.1}$$

where $\theta$ is a smooth, monotone decreasing function, which is positive on the whole space. By setting $\mathbf{q} = \mathbf{r} + t\mathbf{n}$ for some $t \in \Re$, (2.1) can be rewritten as:

$$\sum_{i=1}^{N} \langle \mathbf{n}, \mathbf{p}_i - \mathbf{r} - t\mathbf{n} \rangle^2 \theta\left(\|\mathbf{p}_i - \mathbf{r} - t\mathbf{n}\|\right). \tag{2.2}$$

We define the operator $\mathcal{Q}(\mathbf{r}) = \mathbf{q} = \mathbf{r} + t\mathbf{n}$ as the local minimum of Eq. 2.2 with smallest $t$ and the local tangent plane $H$ near $\mathbf{r}$ accordingly. The local reference domain is then given by an orthonormal coordinate system on $H$ so that $\mathbf{q}$ is the origin of this system. Note that the global minimum of Eq. 2.2 is approaching zero as $t$ approaches $\infty$. To avoid this solution, Eq. 2.2 is normalized with the sum of weights $\theta(\cdot)$.

2. **Local map:** The reference domain for $\mathbf{r}$ is used to compute a local bivariate polynomial approximation to the surface in a neighborhood of $\mathbf{r}$ (see Figure 2.3). Let $\mathbf{q}_i$ be the projection of $\mathbf{p}_i$ onto $H$, and $f_i$ the height of $\mathbf{p}_i$ over H, i.e. $f_i = \mathbf{n} \cdot (\mathbf{p}_i - \mathbf{q})$. Compute the coefficients of a polynomial approximation $g$ so that the weighted least squares error

$$\sum_{i=1}^{N} \left(g(x_i, y_i) - f_i\right)^2 \theta\left(\|\mathbf{p}_i - \mathbf{q}\|\right) \tag{2.3}$$

is minimized. Here $(x_i, y_i)$ is the representation of $\mathbf{q}_i$ in a local coordinate system in $H$. Note that, again, the distances used for the weight function are from the projection of $\mathbf{r}$ onto $H$. The projection $\mathcal{P}$ of $\mathbf{r}$ onto $S_P$ is defined by the polynomial value at the origin, i.e. $\mathcal{P}(\mathbf{r}) = \mathbf{q} + g(0,0)\mathbf{n} = \mathbf{r} + (t + g(0,0))\mathbf{n}$.

### 2.2.2 Properties of the projection procedure

In our application scenario the projection property (i.e. $\mathcal{P}(\mathcal{P}(\mathbf{r})) = \mathcal{P}(\mathbf{r})$) is extremely important: We want to use the above procedure to compute points exactly on the surface. The implication of using a projection operator is that the set of points we project and the order of the points we project do not change the surface. From Eq. 2.2 it is clear that if $(t, \mathbf{n})$ is a minimizer for $\mathbf{r}$ then $(s, \mathbf{n})$ is a minimizer for $\mathbf{r} + (s - t)\mathbf{n}$. Assuming the minimization

is global in a neighborhood $U \in \Re$ of $\mathbf{q}$ then $\mathcal{Q}$ is a projection operation in $U$, because the pair $(0, \mathbf{n})$ is a minimizer for $\mathbf{r} - t\mathbf{n} = \mathbf{q}$. Furthermore, $\mathbf{q} + g(0,0)\mathbf{n} = \mathbf{r} + (t + g(0,0))\mathbf{n}$ is a minimizer and, thus, $\mathcal{P}$ is also a projection in $U$.

We like to stress that the projection property results from using distances to $\mathbf{q}$ rather than $\mathbf{r}$. If $\theta$ depended on $\mathbf{r}$ the procedure would not be a projection because the values of $\theta$ would change along the normal direction.

The surface $S_P$ is formally defined as the subset of all points in $\Re^3$ that project onto themselves. A simple counting argument shows that this subset is a two-parameter family and, thus, $S_P$ a 2-manifold. Eq. 2.2 essentially has $6$ degrees of freedom: $3$ for $\mathbf{r}$, $2$ for $\mathbf{n}$, and $1$ for $t$. On the other hand there are $4$ independent necessary conditions: For a local minimum the partial derivatives of the normal and $t$ have to be zero and for $\mathbf{r}$ to be on the surface $t = 0$ is necessary. This leaves $6 - 4 = 2$ free parameters for the surface. It is clear from simple examples that the parameter family includes manifolds which cannot be expressed as functions over $\Re^2$.

The particular charm of this surface definition is that it avoids piecewise parameterizations. No subset of the surface is parameterized over a planar piece but every single point has its own support plane. This avoids the common problems of piecewise parameterizations for shapes, e.g. parameterization dependence, distortions in the parameterization, and continuity issues along the boundaries of pieces.

However, in this approach we have the problem of proving continuity for any point on the surface because its neighbors have different support planes in general. The intuition for $S_P \in C^\infty$ is, of course, that Eq. 2.1 interpreted as a function $\Re^6 \mapsto \Re$ is $C^\infty$ so that also a particular kernel of its gradient is $C^\infty$.

The approximation of single points is mainly dictated by the weight function $\theta$. The weight function suggested in [Levin 2003] is a Gaussian

$$\theta(d) = e^{-\frac{d^2}{h^2}} \tag{2.4}$$

where $h$ is a fixed parameter reflecting the anticipated spacing between neighboring points. By changing $h$ the surface can be tuned to smooth out features of size $< h$ in $S$. More specifically, a small value for $h$ causes the Gaussian to decay faster and the approximation is more local. Conversely, large values for $h$ result in a more global approximation, smoothing out sharp or oscillatory features of the surface. Figure 2.4 illustrates the effect of different $h$ values.

### 2.2.3 Computing the projection

We explain how to efficiently compute the projection and what values should be chosen for the polynomial degree and $h$. Furthermore, we discuss trade-offs between accuracy and speed.

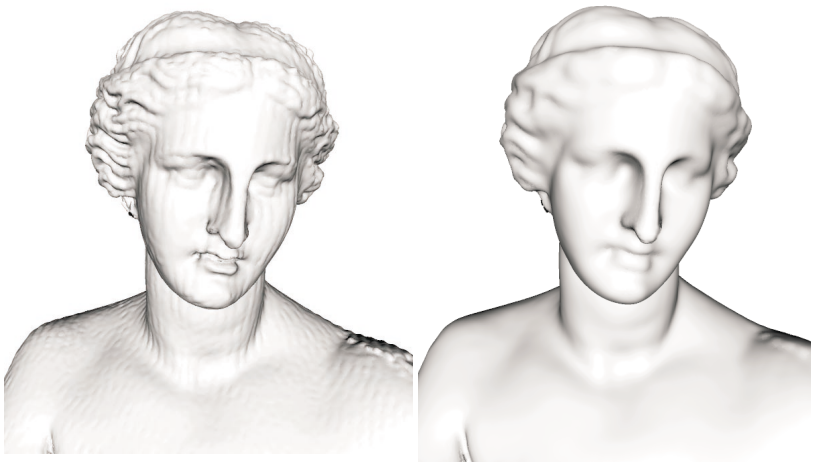


Figure 2.4: The effect of different values for parameter $h$. A point set representing an Aphrodite statue defines an MLS surface. The left side shows an MLS surface resulting from a small value and reveals a surface structure resulting from the wood carving. The right side shows a larger value for $h$, smoothing out small features or noise.

Step 1 of the projection procedure is a non-linear optimization problem. Usually, Eq. 2.2 will have more than one local minimum. By definition, the local minimum with smallest $t$ has to be chosen, which means the plane should be close to $\mathbf{r}$. For minimizing (2.2) we have to use some iterative scheme, which descends toward the next local minimum. Without any additional information we start with t = 0 and first approximate the normal. Note that in this case the weights $\theta_i = \theta(\|\mathbf{p}_i - \mathbf{r}\|)$ are fixed. Let $B = \{b_{jk}\}, B \in \Re^{3\times 3}$ the matrix of weighted covariances be defined by

$$b_{jk} = \sum_i \theta_i \left(\mathbf{p}_{i_j} - \mathbf{r}_j\right)\left(\mathbf{p}_{i_k} - \mathbf{r}_k\right). \tag{2.5}$$

Then, the minimization problem (2.2) can be rewritten in bilinear form

$$\min_{\|\mathbf{n}\|=1} \mathbf{n}^T B \mathbf{n}, \tag{2.6}$$

and the solution of the minimization problem is given as the Eigenvector of $B$ that corresponds to the smallest Eigenvalue.

If a normal is computed (or known in advance) it is fixed and the function is minimized with respect to $t$. This is a non-linear minimization problem in one dimension. In general, it is not solvable with deterministic numerical methods because the number of minima is only bounded by the number N of points $\mathbf{p}_i$. However, in all practical cases we have found that for $t \in [-h/2, h/2]$ there is only one local minimum. This is intuitively clear as $h$ is connected to the feature size and features smaller than $h$ are smoothed out, i.e. features

with distance smaller than $h$ do not provide several minima. For this reason we make the assumption that any point $\mathbf{r}$ to be projected is at most $h/2$ away from its projection. In all our practical applications this is naturally satisfied. Using these prerequisites the local minimum is bracketed with $\pm h/2$. The partial derivative

$$2\sum_{i=1}^{N}\langle \mathbf{n}, \mathbf{p}_i - \mathbf{r} - t\mathbf{n}\rangle \left(1 + \frac{\langle \mathbf{n}, \mathbf{p}_i - \mathbf{r} - t\mathbf{n}\rangle^2}{h^2}\right) e^{-\|\mathbf{p}_i - \mathbf{r} - t\mathbf{n}\|^2/h^2}, \tag{2.7}$$

can be easily evaluated together with the function itself. The derivative is exploited in a simple iterative minimization scheme as explained in [Press et al. 1992, Chapter 10.3].

Once $t \neq 0$ fixing $t$ and minimization with respect to the normal direction is also a non-linear problem because $\mathbf{q} = \mathbf{r} + t\mathbf{n}$ changes and, thus, also the weights change. The search space can be visualized as the tangent planes of a sphere with center point $\mathbf{r}$ and radius $t$. However, in practice we have found the normal (or $\mathbf{q}$) changes only slightly so that we approximate the sphere locally around the current value of $\mathbf{r} + t\mathbf{n}$ as the current plane defined $t$ and $\mathbf{n}$. On this plane a conjugate gradient scheme [Press et al. 1992] is used to minimize among all $\mathbf{q}$ on the plane. The main idea is to fix a subspace for minimization in which $\mathbf{n}$ cannot vanish so that the constraint $\|\mathbf{n}\| = 1$ is always satisfied. The use of a simple linear subspace makes the computation of partial derivatives efficient and, thus, conjugate gradient methods applicable. Clearly, this search space effectively changes $t$ resulting in a theoretically worse convergence behavior. In practice, the difference between the sphere and the plane is small for the region of interest and the effect is not noticeable.

Using these pieces the overall implementation of computing the support plane looks like this:

**Initial normal estimate in $\mathbf{r}$:** The normal in a point might be given as part of the input (e.g. estimate from range images) or could be computed when refining a point set surface (e.g. from close points in the already processed point set). If no normal is available it is computed using the Eigenvector of the matrix of weighted co-variances $B$ (Eq. 2.5).

**Iterative non-linear minimization:** The following two steps are repeated as long as any of the parameters changes more than a predefined $\epsilon$

1. Minimize along $t$, where the minimum is initially bracketed by $t = \pm h/2$,
2. Minimize $\mathbf{q}$ on the current plane $H : (t, \mathbf{n})$ using conjugate gradients. The new value for $\mathbf{q} = \mathbf{r} + t\mathbf{n}$ leads to new values for the pair $(t, \mathbf{n})$.

The last pair $(t, \mathbf{n})$ defines the resulting support plane $H$.

The second step of the projection procedure is a standard linear least squares problem: Once the plane $H$ is computed, the weights $\theta_i = \theta(\|\mathbf{p}_i - \mathbf{q}\|)$ are known. The gradient of

(2.3) over the unknown coefficients of the polynomial leads to a system of linear equations of size equal to the number of coefficients, e.g. 10 for a third degree polynomial.

Through experimentation, we found that high degree polynomials are likely to oscillate. Polynomials of degree 3 to 4 have proved to be very useful as they produce good fits of the neighborhood, do not oscillate, and are quickly computed.

### 2.2.4 Data structures and trade-offs

The most time-consuming step in computing the projection of a point $\mathbf{r}$ is collecting the coefficients from each of the $\mathbf{p}_i$ (i.e. computing the sum). Implemented naively, this process takes $O(N)$ time, where $N$ is the number of points. We exploit the effect of the quickly decreasing weight function in two ways:

1. In a certain distance from $\mathbf{r}$ the weight function is effectively zero. We call this the *neglect distance* $d_n$, which depends solely on $h$. A regular grid with cell size $2d_n$ is used to partition the point set. For each projection operation a maximum of $9$ cells is needed. This results in a very small memory footprint and yields a simple and effective out of core implementation, which makes the storage requirements of this approach independent of the total size of the point set.

2. Using only the selected cells the terms are collected using a hierarchical method inspired by solutions to the N-body problem [Barnes and Hut 1986]. The basic observation is, that a cluster of points far from $\mathbf{r}$ can be combined into one point. To exploit this idea, each cell is organized as an Octree. Leaf nodes contain the $\mathbf{p}_i$'s; inner nodes contain information about the number of points in the subtree and their centroid. Then, terms are collected from the nodes of the Octree. If the node's dimension is much smaller than its distance to $\mathbf{r}$, the centroid is used for computing the coefficients; otherwise the subtree is traversed. In addition, whole nodes can be neglected if their distance to $\mathbf{r}$ is larger than $d_n$.

The idea of neglecting points could be also made independent of numerical issues by using a compactly supported weight function. However, the weight function has to be smooth. An example for such an alternative is

$$\theta(x) = 2x^3 - 3x^2 + 1. \tag{2.8}$$

When Eq. 2.4 and 2.8 are used, the PSS approximates the input set of points. A weight function that interpolates the points should tend to infinity at zero. An example for such a function is:

$$\theta(x) = \frac{e^{-x^2/h^2}}{1 - e^{-x^2/h^2} + \epsilon}\ , \tag{2.9}$$

where $h$ is the same as before, and $\epsilon$ can be a small constant value, or it can be a function of the confidence of the point; interpolating points with high confidence and approximating points with low confidence.

A simple way to trade accuracy for speed is to assume that the plane $H$ passes through the point to be projected. This assumption is reasonable for input points, which are expected to be close to the surface they define (e.g. input that has been smoothed). This simplification saves the cost of the iterative minimization scheme.

## 2.3 Approximation error

Consider again the setting depicted in Figure 2.2. Input points $\{\mathbf{p}_i\}$ define a surface $S_P$, which are then represented by a set of points $\{\mathbf{r}_i\} \in S_P$. However, the set $\{\mathbf{r}_i\}$ defines a surface $S_R$ which approximates $S_P$. Naturally, we would like to have an upper bound on the distance between $S_R$ and $S_P$.

From differential geometry, we know that a smooth surface can be locally represented as a function over a local coordinate system. We also know that in approximating a bivariate function $f$ by a polynomial $g$ of total degree $m$, the approximation error is $\|g - f\| \leq M \cdot h^{m+1}$ [Levin 1998]. The constant $M$ involves the $(m+1)$-th derivatives of $f$, i.e. $M \in O(\|f^{(m+1)}\|)$. In the case of surface approximation, since $S_P$ is infinitely smooth, there exists a constant $M_{m+1}$, involving the $(m+1)$-th derivatives of $S_P$, such that

$$\|S_P - S_R\| \leq M_{m+1} h^{m+1}, \tag{2.10}$$

where $S_R$ is computed using polynomials of degree $m$.

Note that this error bound holds for the MLS surface $S_R$, which is obtained by the projection procedure applied to the data set $R$, using the same parameter $h$ as for $S_P$. Moreover, the same type of approximation order holds for piecewise approximation: Assume each point $\mathbf{r}_i$ defines a support plane so that $S_P$ is a function over a patch $[-h, h]^2$ around $\mathbf{r}_i$ on the plane and, further, that the points $\{\mathbf{r}_i\}$ leave no hole of radius more than $h$ on the surface. Then, the above arguments hold and the approximation error of the union of local, non-conforming, polynomial patches $G_i$ around points $\mathbf{r}_i$ approximates $S_P$ as

$$\left\| S_P - \bigcup G_i \right\| \leq M_{m+1} h^{m+1}. \tag{2.11}$$

Here, $G_i$ is the polynomial patch defined by an MLS polynomial $g_i$ for the point $\mathbf{r}_i$, with corresponding reference plane $H_i$ with normal $\mathbf{n}_i$, a corresponding orthonormal system $\{\mathbf{u}_i, \mathbf{v}_i, \mathbf{n}_i\}$, and an origin at $\mathbf{q}_i$.

$$G_i = \{\mathbf{q}_i + x \cdot \mathbf{u}_i + y \cdot \mathbf{v}_i + g_i(x, y) \cdot \mathbf{n}_i \mid (x, y) \in [-h, h]^2\}. \tag{2.12}$$

These error bounds explain nicely why curvature (see e.g. [Gopi et al. 2000]) is an important criterion for piecewise linear approximations (meshes): The error of piecewise linear functions depends linearly on the second derivatives and the spacing of points. This means, more points are needed where the curvature is higher.

However, when the surface is approximated with higher order polynomials, curvature is irrelevant to the approximation error. Using cubic polynomials, the approximation error depends on the fourth order derivatives of the surface. Note that our visual system cannot sense smoothness beyond second order [Marr 1983]. From that point of view, the sampling density locally depends on an "invisible" criterion. We have found it to be sufficient to fix a spacing $h$ of points $\mathbf{r}_i$ on $S_P$. A smaller $h$ will cause the error to be smaller. However, using higher order polynomials to approximate $S_P$ causes the error to decrease faster when $h$ is reduced.

## 2.4 Generating the representation point set

A given point set might have erroneous point locations (i.e. is noisy), may contain too many points (i.e. is redundant) or not enough points (i.e. is under-sampled).

The problem of noise is handled by projecting the points onto the MLS surface they define. The result of the projection procedure is a thin point set. Redundancy is avoided by decimating the point set, taking care that it persists to be a good approximation of the MLS surface defined by the original point set. In the case of under-sampling, the input point set needs to be up-sampled. In the following sections, we show techniques to remove and add points.

Figure 2.5 illustrates the idea of resampling at the example of the Buddha statue. Using the techniques presented below it is possible to resample the geometry to be evenly sampled on the surface.

### 2.4.1 Down-sampling

Given a point set, the decimation process repeatedly removes the point that contributes the smallest amount of information to the shape. The contribution of a point to the shape or the error of the sampling is dictated by the definition of the shape. If the point set is reconstructed by means of a triangulation, criteria from the specific triangulation algorithm should control the resampling. Criteria include the distance of points on the surface [Hoppe et al. 1992], curvature [Gopi et al. 2000], or distance from the medial axis of the shape [Amenta et al. 1998]. For a direct display of the point set on a screen homogeneous distribution of the points over the surface is required [Grossman and Dally 1998; Pfister et al. 2000]. Here, we derive a criterion motivated by our definition of the surface.

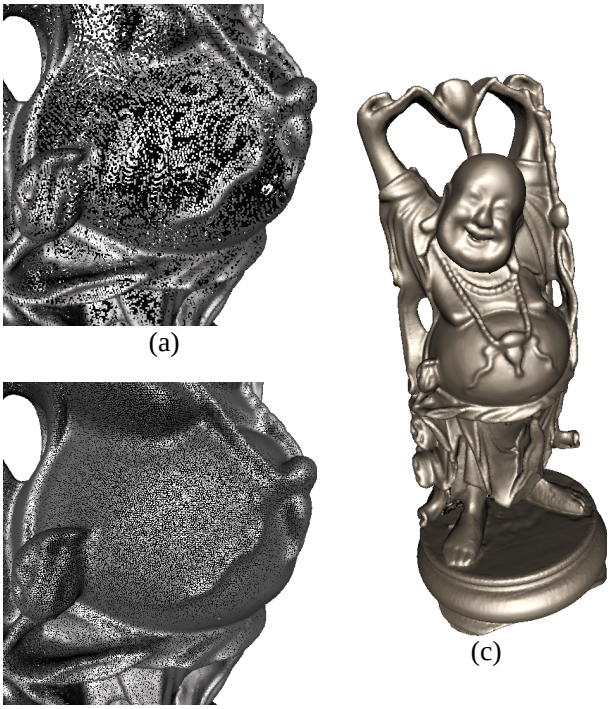



Figure 2.5: Points acquired by range scanning devices or vertices from a processed mesh typically have uneven sampling density on the surface (a). The sampling techniques discussed here allow to evenly resample the object (b) to ensure a sufficient density for further processing steps, for example rendering (c).

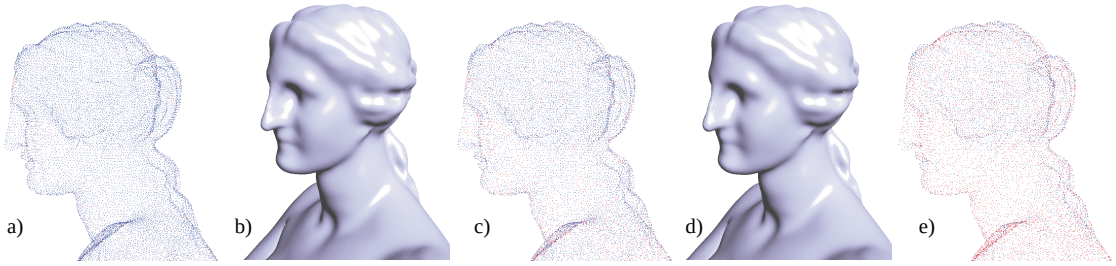



Figure 2.6: The point set representing an Aphrodite statue is projected onto a smooth MLS-surface. After removing redundant points, a set of 37K points represents the statue (a). The corresponding rendering is shown in (b). The point set is decimated using point removal. An intermediate stage of the reduction process is shown in (c). Note that the points are color-coded with respect to their importance. Blue points do not contribute much to the shape and might be removed; red points are important for the definition of the shape. The final point set in (e) contains 20K points. The corresponding rendering is depicted in (d) and is visually close to the one in (b).

The contribution of a projected point $\mathbf{p}_i$ to the surface $S_P$ can be estimated by comparing $S_P$ with $S_{P-\{\mathbf{p}_i\}}$. Computing the Hausdorff distance between both surfaces is expensive and not suitable for an iterative removal of points of a large point set. Instead, we approximate the contribution of $\mathbf{p}_i$ by its distance from its projection onto the surface $S_{P-\{\mathbf{p}_i\}}$. Thus, we estimate the difference of $S_P$ and $S_{P-\{\mathbf{p}_i\}}$ by projecting $\mathbf{p}_i$ onto $S_{P-\{\mathbf{p}_i\}}$ (projecting $\mathbf{p}_i$ under the assumption it was not part of $P$).

The contribution values of all points are inserted into a priority queue. At each step of the decimation process, the point with the smallest error is removed from the point set and from the priority queue. After the removal of a point, the error values of nearby points have to be recalculated since they might have been affected by the removal. This process is repeated until the desired number of points is reached or the contributions of all points exceed some prespecified bound.

Figure 2.7 illustrates our decimation process applied on the set of red points depicted in (a). First, the red points are projected on the MLS surface to yield the blue points. A close-up view over a part of the points shows the relation between the input (red) points and the projected points. In (b), we show three snapshots of the decimation process, where points are colored according to their error value; blue represents low error and red represents high error. Note that in the sparsest set, all of the points have a high error, that is, their removal will cause a large error. As the decimation proceeds, fewer points remain and their importance grows and the error associated with them is larger. Figure 2.6 shows the decimation process in 3D with corresponding renderings of the point sets.

### 2.4.2 Up-sampling

In some cases, the density of the point set might not be sufficient for the intended usage (e.g. direct point rendering or piecewise reconstructions). To alleviate this problem, we try to decrease the spacing among points. Additional points should be placed (and then projected to the MLS surface) where the spacing among points is larger then a specified bound.

The basic idea of our approach is to compute Voronoi diagrams on the MLS surface and add points at vertices of this diagram. Note that the vertices of the Voronoi diagram are exactly those points on the surface with maximum distance to several of the existing points. This idea is related to Lloyd's method [Lloyd 1982], i.e techniques using Voronoi diagrams to achieve a certain distribution of points [Okabe et al. 1992]. The same idea was used for image by Eldar et al. [1997].

However, computing the Voronoi diagram on the MLS surface is excessive and local approximations are used instead. More specifically, our technique works as follows: In each step, one of the existing points is selected randomly. A local linear approximation is built and nearby points are projected onto this plane. The Voronoi diagram of these points

is computed. Each Voronoi vertex is the center of a circle that touches three or more of the points without including any point. The circle with largest radius is chosen and its center is projected to the MLS surface. The process is repeated iteratively until the radius of the largest circle is less than a user-specified threshold. (see Figure 2.8). At the end of the process, the density of points is locally near-uniform on the surface. Figure 2.8 shows the original sparse point set containing 800 points, and the same object after adding 20K points

## 2.5 Rendering

The challenge of our interactive point rendering approach is to use the representation points and (when necessary) create new points by sampling the implicitly defined surface at a resolution sufficient to conform to the screen space resolution (see Figure 2.9 for an illustration of that approach).

Usually, the representation points are not sufficient to render the object in screen space. In some regions, it is not necessary to render all points as they are occluded, backfacing, or have higher density than needed. However, typically, points are not dense enough to be projected directly as a single pixel and more points need to be generated by interpolation in object space.

### 2.5.1 Culling and view dependency

The structure of our rendering system is similar to QSplat [Rusinkiewicz and Levoy 2000]. The input points are arranged into a bounding sphere hierarchy. For each node, we store a position, a radius, a normal coverage, and optionally a color. The leaf nodes additionally store the orientation of the support plane and the coefficients of the associated polynomial. The hierarchy is used to cull the nodes with the view frustum and to apply a hierarchical backface culling [Kumar et al. 1996]. Note that culling is important for our approach since the cost of rendering the leaf nodes (evaluating the polynomials) is high compared to simpler primitives. Moreover, if the traversal reaches a node with an extent that projects to a size of less than a pixel, this node is simply projected to the frame-buffer without traversing its subtree. When the traversal reaches a leaf node and the extent of its bounding sphere projects to more than one pixel in screen space, additional points have to be generated.

The radius of a bounding sphere in leaf nodes is simply set to the anticipated feature size $h$. The bounding spheres for inner nodes naturally bound all of their subtree's bounding spheres. To compute the size of a bounding sphere in pixel space the radius is scaled by the model-view transform and then divided by the distance of center and eye point in z-direction to accommodate the projective transform.

### 2.5.2 Sampling additional points

The basic idea is to generate a grid of points sufficient to cover the extent of a leaf node. However, projecting the grid points using the method described in Section 2.2 is too slow in interactive applications. The main idea of our interactive rendering approach is to sample the polynomials associated with the representation points rather than really projecting the points.

It is clear that the union of these polynomial patches is not a continuous surface. However, the Hausdorff-error of this approximation is not worse than the error of the surface computed by projecting every point using the operator $\mathcal{P}$ (see Section 2.3). Since the Hausdorff-error in object space limits the screen space error the error bound can be used to make the approximation error invisible in the resulting image.

However, to conform with the requirements formulated in Section 2.3 the point set and the associated polynomials are required to be near-uniform on the surface. It might be necessary to first process a given point set with the up-sampling methods presented in Section 2.4. This way, we ensure that the local, non-conforming (i.e. overlapping or intersecting) polynomials are a good approximation to the surface inside a patch $[-h, h]^2$ around a point and, thus, the resulting image shows a smooth surface. However, most dense point sets can be readily displayed with the approach presented here. For example, Figure 2.11 shows several renderings of the original Stanford Bunny data.

It is critical to properly define the extent of a polynomial patch on the supporting plane, such that neighboring patches are guaranteed to overlap (to avoid holes) but do not overlap more than necessary. Since no inter-point connectivity information is available, it is unclear which points are immediate neighbors of a given point on the surface.

To compute the extent of a polynomial patch associated to the point $\mathbf{p}_i$ on the support plane $H$ all points inside a ball of radius $h$ around the projection $\mathbf{q} = \mathcal{Q}(\mathbf{p}_i)$ are collected. These points are projected to the support plane $H$, which leads to local $(u, v)$ coordinates for each projected point. The extent is defined by a circle around $\mathbf{q}$ that encloses all projected points. More specifically, assume $\mathbf{q}$ has the local coordinate $(0, 0)$, the radius of the circle is given by the largest 2-norm of all local $(u, v)$ coordinates of projected points. Since the spacing of points is expected to be less than $h$, patches of neighboring points are guaranteed to overlap.

Note that using a constant extent (e.g. a disk of radius $h$ on the support plane) can lead to errors, as the polynomial $g$ over $H$ might leave the ball of radius $h$, in which a good approximation of the point set is expected. Figure 2.10 illustrates the computation of the patch sizes.

The grid spacing $d$ should be so computed that neighboring points have a screen space distance of less than a pixel. Thus, the grid spacing depends on the orientation of the polynomial patch with respect to the screen. Since the normals change on each polynomial

patch we rather use a simple heuristic to conservatively estimate $d$: The grid spacing $d$ is computed so that a grid perpendicular to the viewing direction is sufficiently sampled in image space.

If the grid is, indeed, perpendicular to the viewing direction, the sampling is also correct on the polynomial. If the grid is not perpendicular to the viewing direction, the projected area might be oversampled or undersampled depending on the orientation of the support plane and the derivatives of the polynomial. Note, however, that sufficient sampling is guaranteed if the derivatives are less than $1$. In practice we have rarely encountered higher derivatives so we decided not to evaluate the maximum derivatives of all polynomial patches. However, this could be done in a preprocess and the density could be adjusted accordingly.

Upon the view-dependent grid spacing $d$, the polynomials are evaluated by a forward difference approach, where the polynomial is scanned across its extent in its local $u, v$ parametric space. The affine map transforming from support plane coordinates to world coordinates is factored into polynomial evaluation, thus, generating points in world coordinates. These points are then fed into the graphics pipeline to be projected to the screen.

Surprisingly, we have found that quad meshes are processed faster by current graphics hardware than a set of points. For this reason we use quad meshes to represent the polynomial patches. This has the additional advantage that during lazy evaluation (see below) no holes occur.

### 2.5.3 Grid pyramids

The time-critical factor is the view-dependent evaluation of the points on the polynomial. Optimally, these are recomputed whenever the projected screen space size changes. To accelerate the rendering process, we store a grid pyramid with various resolutions per point. Initially, the pyramid levels are created, but no grid is actually evaluated. When a specific grid resolution is needed, the system creates and stores the level that slightly oversamples the polynomial for a specific resolution, such that small changes in the viewing position do not result in new evaluations.

To enhance the interactivity of our approach, we also allow the point size to adapt to changing viewing conditions. For example, while rotating or zooming, sparser grids with large points are used to guarantee an interactive frame rate. Once the viewer stops moving, a proper grid is chosen from the pyramid.

### 2.5.4 Results

We have tested our approach on a variety of point sets. Figure 2.11 shows the renderings of the Stanford Bunny. In (a), the original point set is shown. Splatting (b) is not leading to

good results, because the model is not sampled densely enough. The traditional Gouraud-shaded mesh in (c) and (g) is compared to our approach in (d) and (h). Note the accuracy of the highlights. The non-conforming local polynomial patches are shown color-coded in (e) and (f). An example of a environment mapping to demonstrate the normal continuity is given in Figure 2.12. Note that silhouettes and normals are smooth, which leads to less distortions on the boundary and in the reflections.

The frame rates we achieve are mainly dictated by the number of visible representation points (i.e. graph traversal time) and the number of pixels to be filled. All models depicted here are displayed at more than 5 frames per second in a $512^2$ screen window (see the accompanying video for more information). The number of representation points ranges from 1000 (for the torus) to 900K (for the angel statue). Tests are performed on a PC with GeForce2 graphics board.

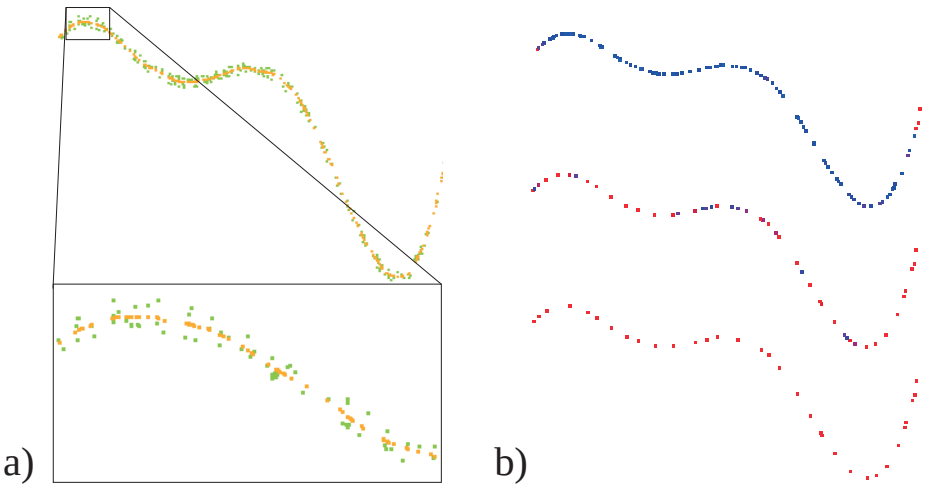



Figure 2.7: Noisy input points (green points) are projected onto their smooth MLS curve (orange points). The figures in (a) show the point sets and a close-up view. The decimation process is shown in (b). Points are color-coded as in Figure 2.6.

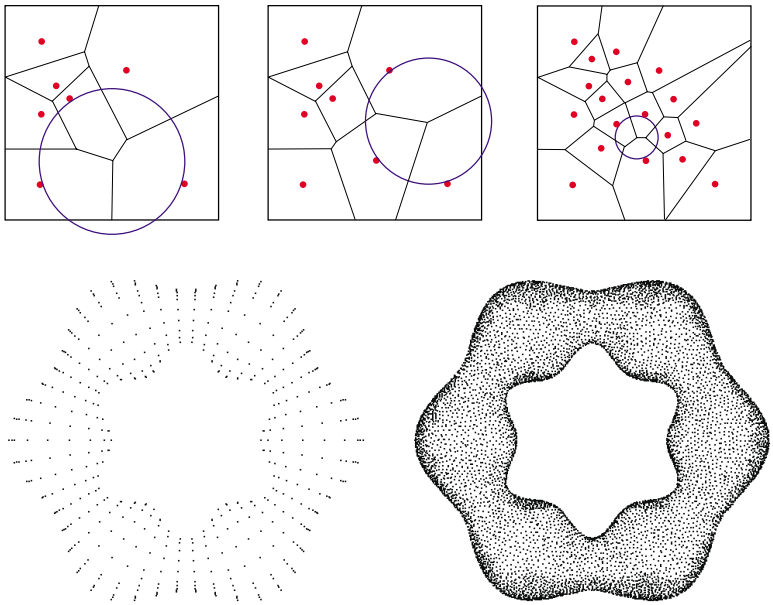


Figure 2.8: The up-sampling process: Points are added at vertices of the Voronoi diagram. In each step, the vertex with the largest empty circle is chosen. The process is repeated until the radius of the largest circle is smaller than a specified bound. The wavy torus originally consisting of 800 points has been up-sampled to 20K points.

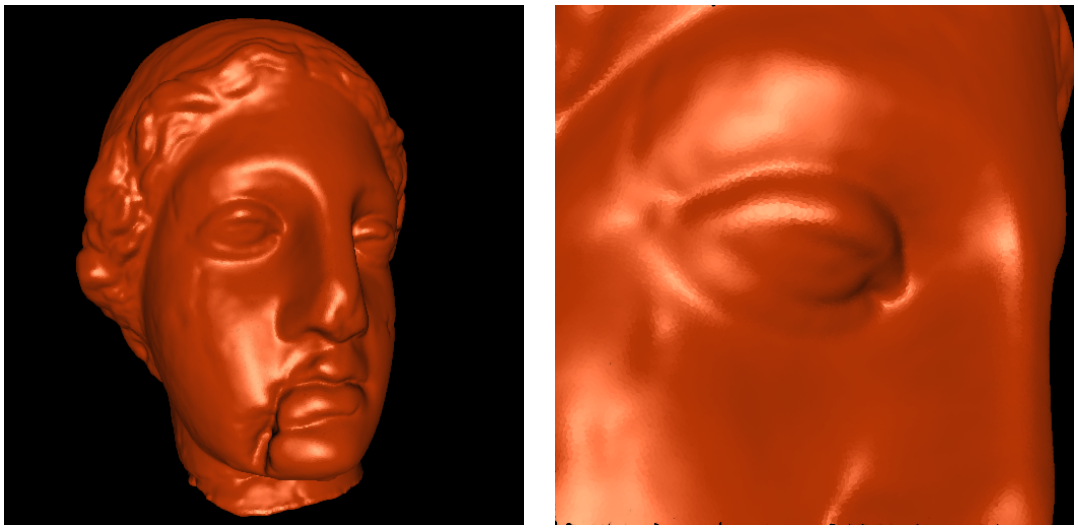


Figure 2.9: Up-sampling could be used generate enough points on the surface to conform with the resolution of the image to be rendered. The right image shows a close-up rendered with splats.

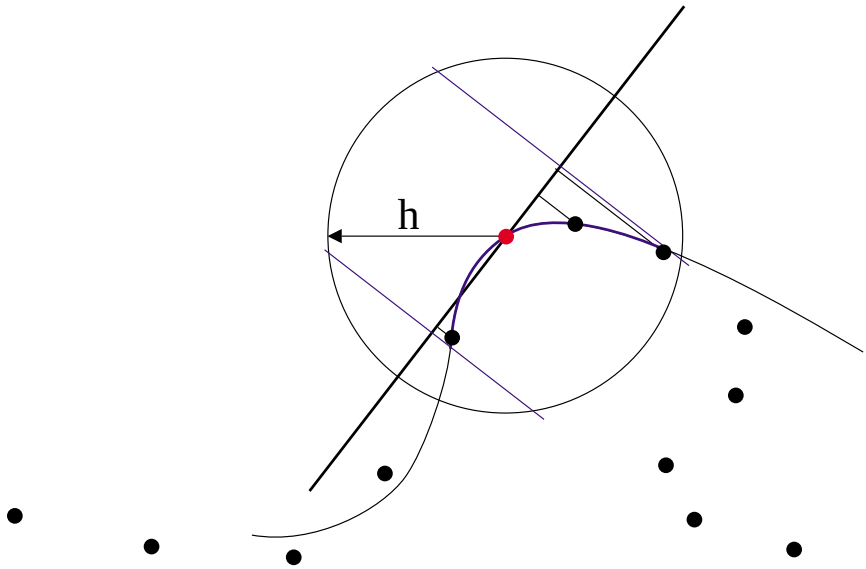



Figure 2.10: The patch size of a polynomial: Points inside a ball of radius $h$ around the red point are projected onto the support plane of the red point. The patch size is defined as the bounding box (in local coordinates) of the projections. Note that using a disk of radius $h$ or a square patch of $[-h, h]^2$ would lead to unpleasant effects in some cases, as the polynomial might leave the ball of radius $h$.

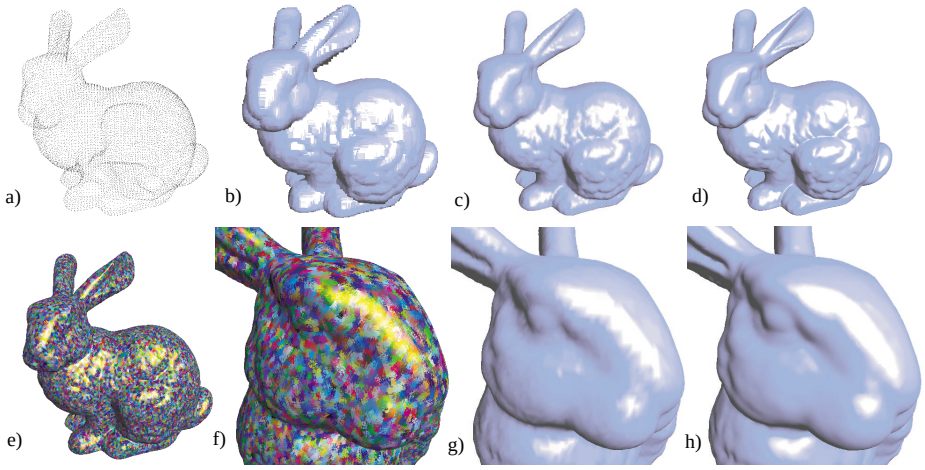



Figure 2.11: The Stanford Bunny: The points defining the bunny are depicted in (a) (some points are culled). Points are splatted in (b) to satisfy screen space resolution. Note the difference of a piecewise linear mesh over the points (c) and close-up in (g) to the rendering of non-conforming polynomial patches (d) and (h). The patches are color-coded in (e) and (f).

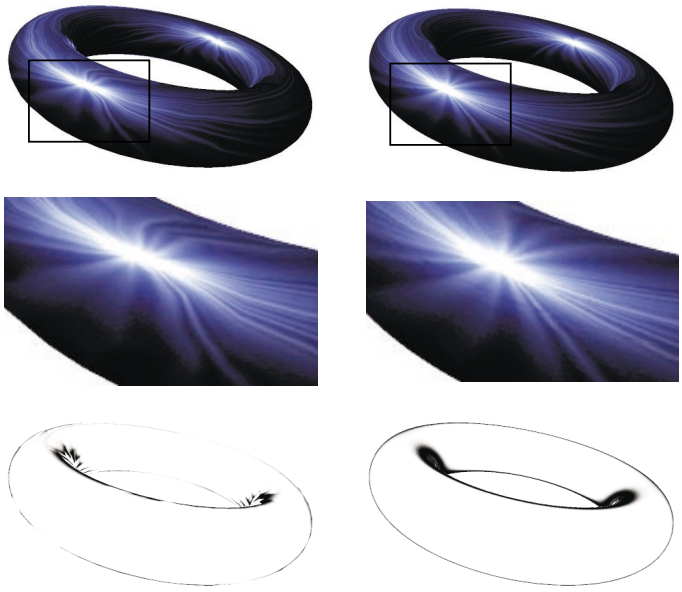


Figure 2.12: Comparison of mesh rendering with our technique with environment mapping. The left column shows renderings of a mesh consisting of 1000 vertices. The right column shows our technique using the vertices as input points. The environment maps emphasize the improved normal and boundary continuity.

# Chapter 3

# Progressive Point Set Surfaces

Point sets are emerging as a surface representation. The particular appeal of point sets is their generality: every shape can be represented by a set of points on its boundary, where the degree of accuracy typically depends only on the number of points. Point sets do not have a fixed continuity class or are limited to certain homology groups as in most other surface representations. Polygonal meshes, in particular, have a piecewise linear $C^0$ geometry, resulting in an unnatural appearance. To overcome the continuity problem, research has been devoted to image space smoothing techniques (e.g. Gouraud shading), or procedures to smooth the model's geometry such as subdivision surfaces.

To define a manifold from the set of points, the inherent spatial interrelation among the points is exploited as implicit connectivity information. A mathematical definition or algorithm attaches a topology and a geometric shape to the set of points. This is non-trivial since it is unclear what spacing of points represents connected or disconnected pieces of the surface. Moreover, most surfaces are manifold, which limits the possibilities of using functions for global interpolation or approximation. Recently, Levin gave a definition of a manifold surface from a set of points [Levin 2003], which was used in [Alexa et al. 2001] to render shapes.

To achieve a certain geometric fidelity many points are needed to describe a shape. The necessary uniformity of the point's density might further increase the number of points. The relation between point density and accuracy calls for the definition of levels of detail and the notion of progressiveness. That is, the point set should have a base set that represents a coarse and a smooth version of the surface, which can be refined by a series of point insertions in the spirit of progressive meshes [Hoppe 1996], or adaptive parameterization of surfaces (MAPS) [Lee et al. 1998; Guskov et al. 2000].

Progressive or multi-scale representations are useful not only to cut down on the amount of data but also for modelling and visualization purposes (See Figure 3.1). This is because of the connection between detail levels and spectral bands [Kobbelt et al. 1998a; Kobbelt et al. 1998b; Zorin et al. 1997]. Ideally, the geometric representations of the levels are

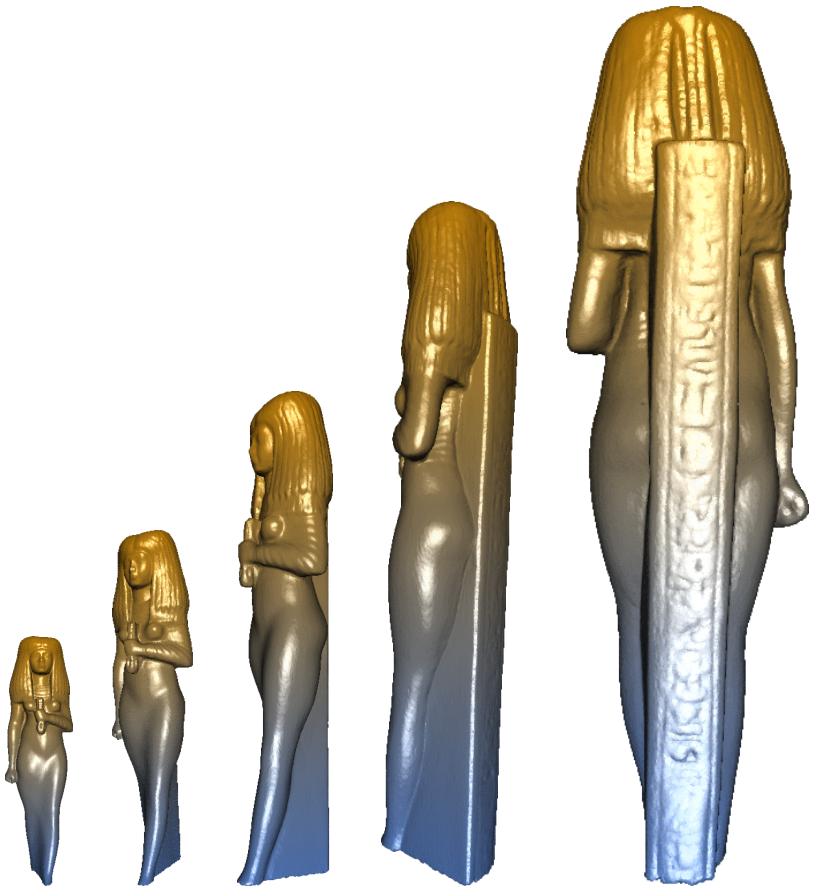


Figure 3.1: The progressive series of the Isis model, on the left a model with 19K points progressively refined up to 189K points in the model on the right.

relative to each other and independent of the position and orientation of the shape. This is achieved by decomposing the geometric components into a normal and a tangential direction, and encoding each level as a displacement of a coarser level [Khodakovsky et al. 2000; Guskov et al. 2000]. Furthermore, tangential components can be described implicitly by the refinement rule so that a single scalar / point is sufficient to encode the shape. Note that this also leads to an efficient geometry compression scheme.

This approach has been described and analyzed for mesh geometry using subdivision techniques by [Guskov et al. 2000; Lee et al. 2000]. Based on the method of moving least squares (MLS) presented in [Alexa et al. 2001; Levin 2003], in this paper we define a projection operator and a refinement rule. Together, they allow us to refine a given base point set towards a reference point set (input model). The projection operator defines a local tangential coordinate frame, which allows us to specify the position of inserted points, with a scalar representing the normal component. The tangential components are defined by the

refinement rule. As such, the scheme is reminiscent of subdivision techniques for meshes.

Based on the properties of the refinement process we develop a simplification scheme for point sets to construct a base point set, which represents a smoother version of the original shape. The base point set is then refined by point insertion to add levels of detail. The surface could be refined adaptively creating point sets with densities varying with respect, say, to the viewing parameters or the local geometric behavior of the surface. In this paper we:

- introduce a point-based geometric modelling technique that is based on the MLS projection mechanism;
- present a progressive scheme for point set surfaces, where the levels of an object represent both coarse–to–fine and smooth–to–detailed hierarchy;
- use a local operator that allows accurate computation of local differential surface properties;
- apply an encoding scheme for compressing progressive point sets.

## 3.1 Related Work

Our work is related to the recent research efforts in developing point-based representations for shapes [Grossman and Dally 1998; Pauly and Gross 2001; Pfister et al. 2000; Rusinkiewicz and Levoy 2000]. Most of the mentioned techniques are targeted at fast processing and rendering of large point-sampled geometry. Our techniques are focused on advancing the "modelling" of primitives with points. In this respect our work fits into the field of *digital geometry processing* (DGP) [Desbrun et al. 1999; Guskov et al. 1999; Guskov et al. 2000; Lee et al. 1998; Taubin 1995], which aims at extending standard signal processing concepts to the manifold surface domain. [Pauly and Gross 2001], have shown how to extend these techniques to point-based representations by collecting sets of points to patches, over which the points define an irregularly sampled function. Most approaches for meshes also construct a multiresolution representation by progressively refining a base domain and exploiting the connection of the refinement levels to spectral properties. Meshes are generally composed of two parts: the connectivity of the mesh; and the geometry (i.e., the position of the vertices). Few DGP techniques can be directly applied to such a representation (one example is the pioneering work presented in [Taubin 1995]). DGP algorithms require parameterization of the surface, which could be represented in mesh form as a subdivision surface [Lee et al. 1998; Kobbelt et al. 1999; Eck et al. 1995].

A recent work related to our surface representation is [Carr et al. 2001] which reconstructs 3D objects by fitting a global radial basis function (RBF) to point clouds. An RBF

forms a solid model of an object, allowing analytic evaluation of surface normal, direct rendering and iso-surface extraction, similar to the properties of the surface representation we use.

Several different hierarchical representations have been proposed for geometric objects. Object simplification [Cignoni et al. 1994; Cohen et al. 1996; He et al. 1996; Hoppe et al. 1993; Zhou et al. 1997] is often used to generate a hierarchical representation, which could be used for many purposes, for example, rendering [Duchaineau et al. 1997; El-Sana and Varshney 1999; Xia et al. 1997].

Linsen [2001] also describes a multiresolution representation of point-based objects. Similarly to our method, the detail points are inserted using a prediction operator. In our work we focus on a space efficient progressive representation of a point set. Another recent related work is the one by Pauly et al. [2002], which describes a number of point-based simplification methods. The method we present here for building the base point set is similar to their clustering method.

A leading technique for representing hierarchical meshes is *progressive meshes* [Hoppe 1996], a mesh representation of varying resolution where a series of edge-split operations progressively refines a base mesh up to the original resolution of the source. This representation motivates solutions to mesh simplification, progressive transmission and loading from a disk or from a remote server. A number of mesh compression and streaming techniques are based on this concept [Cohen-Or et al. 1999; Pajarola and Rossignac 2000; Taubin et al. 1998]. For a recent survey of mesh simplification and compression techniques see [Gotsman et al. 2001].

Subdivision surfaces [Catmull and Clark 1978; Warren and Weimer 2001] are defined by a topological refinement operator and a smoothing rule. Given a mesh of arbitrary topology, they refine the mesh toward a smooth limit surface. Point set surfaces are similar to subdivision surfaces, in that it is possible to add points to the defined smooth surface without additional information. However, to describe an arbitrary surface using subdivision techniques, inserted points need to be displaced. Normal Meshes [Guskov et al. 2000] as well as *displaced subdivision surfaces* [Lee et al. 2000] demonstrate this idea of a multiresolution subdivision mesh where vertices are displaced by a single scalar value in the normal direction. These approaches are attractive since a single scalar value is easier to manipulate or store. The underlying concept can be understood as decomposing the surface representation into a tangential and a normal component (see [Guskov et al. 2000]). Note that we use the same idea, however without coding the tangential component explicitly as the mesh connectivity but implicitly as point proximity.

## 3.2 MLS surfaces

The MLS surface $S_P$ of a set of points $P = \{\mathbf{p_i}\}, \mathbf{p_i} \in \Re^3, i \in \{1, \ldots, N\}$ is defined implicitly by a projection operator. To project a point $\mathbf{r}$ onto $S_P$ two steps are necessary: First, a local reference domain $H = (\mathbf{n}, d)$, where $d$ is the origin of the plane and $\mathbf{n}$ is the normal to the plane is computed. Then, a local bivariate polynomial is fitted over $H$ to the point set. More precisely, the local reference domain $H = \{\mathbf{x} | \langle \mathbf{n}, \mathbf{x} \rangle - \mathbf{d} = 0, \mathbf{x} \in \Re^3\}, \mathbf{n} \in \Re^3, \|\mathbf{n}\| = 1$ is determined by minimizing

$$\sum_{i=1}^{N} \left( \langle \mathbf{n}, \mathbf{p_i} - \mathbf{r} - t\mathbf{n} \rangle \right)^2 e^{-\|\mathbf{p_i} - \mathbf{r} - t\mathbf{n}\|^2 / h^2} \tag{3.1}$$

in all normal directions $\mathbf{n}$ and offsets $t$. Here $d = \mathbf{r} + t\mathbf{n}$. A local coordinate system over $H$ is defined by taking the standard basis for $\Re^3$, $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$ and compute a rotation matrix $M$ such that $\mathbf{n} = M \cdot \mathbf{e_3}$. The local coordinate system is defined by $(M \cdot \mathbf{e_1}, M \cdot \mathbf{e_2})$. This rotation matrix is not uniquely defined. For our purposes, any solution will do as long as we use the same procedure to compute $M$ in all of our computations.

Let $\mathbf{q}$ be the projection of $\mathbf{p_i}$ onto $H$, and $f_i$ the height of $\mathbf{p_i}$ over $H$, i.e. $f_i = \mathbf{n} \cdot (\mathbf{p_i} - \mathbf{q})$. The polynomial approximation $g$ is computed by minimizing the weighted least-squares error

$$\sum_{i=1}^{N} \left( g(x_i, y_i) - f_i \right)^2 e^{-\|\mathbf{p_i} - \mathbf{r} - t\mathbf{n}\|^2 / h^2}. \tag{3.2}$$

The projection of $\mathbf{r}$ is given by

$$MLS(\mathbf{r}) = \mathbf{r} + (t + g(0, 0))\mathbf{n}. \tag{3.3}$$

Formally, the surface $S_P$ is the set of points ($\Omega \subset \Re$) that project onto themselves. In this definition, $h$ is the anticipated spacing of the points. Thus, the point set together with an adequately chosen value for $h$ defines the surface. As part of the projection procedure, we not only determine the position on the surface where a point projects to, but we also obtain high-order derivative information analytically, which can be used to accurately determine normal, curvature, etc. A detailed description of computing the reference domain and polynomial is presented in [Alexa et al. 2003]. In the following, $S_X$ will generally be the surface with respect to a point set $X$ defined by the above procedure. We will call this the MLS-surface of $X$.

## 3.3 Progressive point set surfaces

First, we give an overview of the concept of progressive point set surfaces (PPSS) and then explain the details in the following sections.

Given a reference (input) point set $R = \{\mathbf{r_i}\}$ defining a reference surface $S_R$. The point set $R$ is reduced by removing points to form a base point set $P_0 \subset R$. This base point set defines a surface $S_{P_0}$ which differs from $S_R$. Next, we will resample the surface adding more points so that the difference between our surface and $S_R$ decreases.

The base point set $P_0$ is refined by inserting additional points yielding the set $P_1$. The refinement operator first inserts points independent of the reference set $R$, which means $P_1 \not\subset R$. Then, the inserted points are displaced so that the difference between the surfaces decreases, i.e. $d(S_{P_1}, S_R) < d(S_{P_0}, S_R)$.

This process is repeated to generate a sequence of point sets $P_i$ with increasing size and decreasing difference from the reference surface. A progressive point set surfaces is defined as the MLS surface of $\mathcal{P} = P_0, P_1, \ldots$, where each point set $P_i$ is encoded by the (scalar) displacements of inserted points, yielding a compact representation of the progressive point set surface.

In the following we explain the necessary steps for this procedure in detail. We denote the reference domain plane of a point $p$ with respect to a point set $S$ as $H_S(p)$ and the polynomial is similarly denoted by $g_S(p)$.

### 3.3.1 The refinement operator

Let $R$ be the reference point set as before and $P$ be the point set to refine. The set $P$ is refined by generating additional points $A = \{\mathbf{a_j}\}$, which are sampled in the local neighborhoods of the $\mathbf{p_i}$.

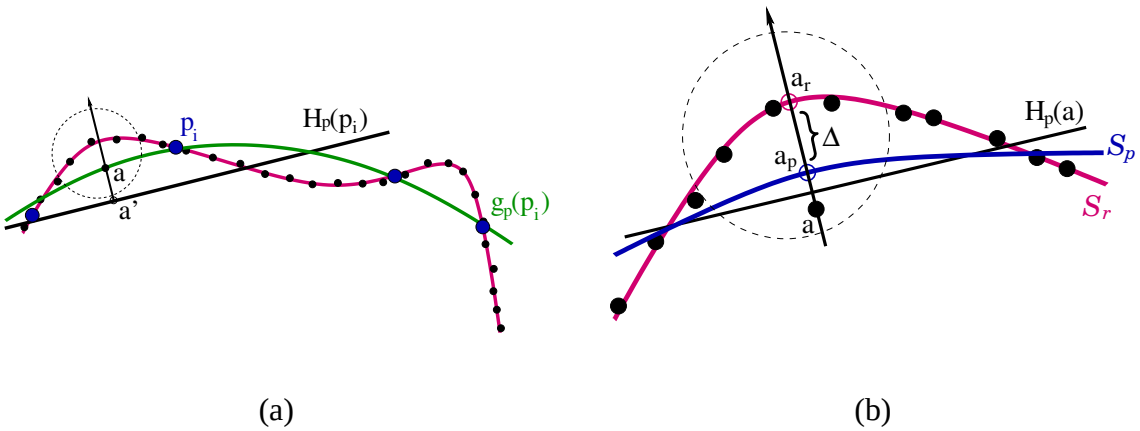
More specifically, let the local reference domain $H_p(\mathbf{p_i})$ of a point $\mathbf{p_i}$ (see Figure 3.2a) be determined as the local minimum of Eq. (3.1) with respect to the points in $P$. The reference plane $H_p(\mathbf{p_i})$ is sampled regularly at intervals $\rho$, yielding a set of $(u, v)$ coordinates for the points $\mathbf{a'_j}$ on the plane. These points are placed in the neighborhood of the surface using the polynomial $\overline{g} = g_p(\mathbf{p_i})$ (defined by Eq. (3.2) ), yielding the set of points $\mathbf{a_j} = (u, v, \overline{g}(u, v))$.

To find and encode the positions of the additional points, the plane $H_p(\mathbf{a_j})$ of $\mathbf{a_j}$ is computed, and then two local polynomial fits are computed on the basis of the given reference domain $H_p(\mathbf{a_j})$: the first polynomial $g_p$ is with respect to the points in $P$ and the second, $g_r$, is with respect to the points in $R$. The height of a point $\mathbf{a_j}$ is given as $g_r(0, 0)$, i.e. on the local polynomial fit to the points in the reference set (see Figure 3.2b).

Since the coordinate $(u, v)$ is generated implicitly by specifying the regular sampling parameter $\rho$, only the height has to be encoded. Based on Eq. (3.3), the height of $\mathbf{a_j}$ can be expressed as the difference:

$$\Delta = g_r(0, 0) - g_p(0, 0). \tag{3.4}$$

Since the distance between $g_r$ and $g_p$ is significantly smaller than the distance between $g_r$ and the reference plane, the $\Delta$ can be efficiently encoded.




(a) (b)

Figure 3.2: An illustration of the point set refinement process: (a) a new point $a$ is generated in the neighborhood of the surface of the current level (the blue points). The reference plane $H_p(p_i)$ and polynomial $\overline{g} = g_p(p_i)$ of $p_i$ are computed. By scanning the neighborhood of $p_i$, a new point $a'$ on $H_p(p_i)$ is generated and projected on the polynomial $\overline{g}$, i.e. $a = \overline{g}(a')$. In (b), $a$ is projected on $MLS_p$ (the blue curve) by computing its reference plane $H_p(a)$ and polynomial $g_p(a)$. Next, $a$ is projected again on $S_R$ (defined by the black points) using the same reference plane, but with the appropriate polynomial $g_r(a)$. Finally, the detail value $\Delta = a_r - a_p$ is computed.

The regular sampling pattern has to be adapted to avoid oversampling and sampling of empty regions. We introduce two criteria for deciding whether to insert a point. A point $\mathbf{a_i}$ is inserted only if

1. none of the already inserted points $P \cup \{\mathbf{a_j}, j < i\}$ is closer than $\rho$ and
2. it is in the convex hull of points in $P$ closer than $h$ or, more formally, iff $\mathbf{a_i} \in \mathcal{CH}\{\mathbf{p_i} | \|\mathbf{p_i} - \mathbf{a_i}\| < h\}$.

The first criterion avoids oversampling, and the second aims at detecting boundaries of the manifold by defining the extent of the local neighborhood.

The sampling parameter $\rho$ can be used to specify the refinement convergence. If $\rho$ is halved in every refinement step the number of points approximately quadruples from one point set to the next. However, $\rho$ can also be used to adapt the sampling density to the application needs, for example, visible regions of the surface or local curvature if piecewise linear approximations are needed.

As the point density increases from one refinement level to the next, also $h$ should be adapted. We adapt $h$ to the change in $\rho$, for example, if $\rho$ is halved in every step, so is $h$. We assume, however, that a suitable $h$ is given for the base point set. This is discussed in the following section.

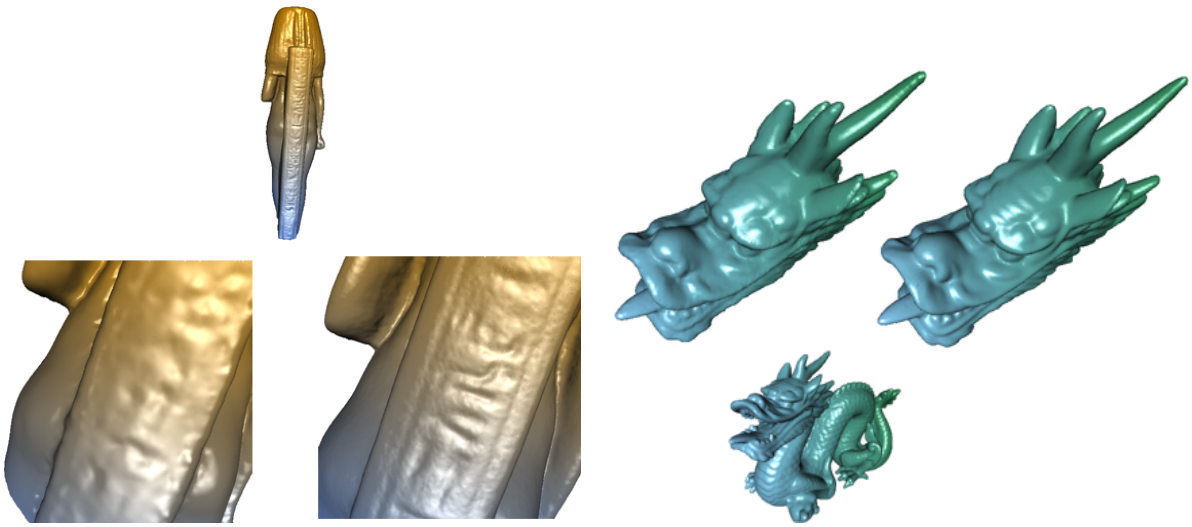


Figure 3.3: Close-ups of smooth renderings for different levels in the hierarchy of a PPSS. The base level of a PPSS defines a smooth surface, which is visualized here by upsampling the smooth surface by adding points without displacing them. Note that higher levels add the missing details to the PPSS representation.

### 3.3.2 Constructing the base point set

The refinement operator uses local reference domains on the basis of the reduced point set to compute polynomial fits to the reference point set. This requires the local reference domain of a point $\mathbf{p_i}$ with respect to the points in $P$ and to the points in $R$ to be about the same. We use this requirement as a criterion for reducing the reference point set to the base point set.

Given a neighborhood size $h$ and a maximum deviation $\epsilon$, let $Q_i$ be the point set which results from removing the points in a $h$-neighborhood around $\mathbf{r_i}$, i.e.

$$Q_i = \{\mathbf{r_k} | \|\mathbf{r_k} - \mathbf{r_i}\| > h\}. \tag{3.5}$$

A point $\mathbf{r_i}$ can be used in the base point set if its original reference domain $H_R(\mathbf{r_i})$ is close to the reference domain $H_{Q_i}(\mathbf{r_i})$ with respect to the reduced point set $Q_i$. More specifically, the distance between $H_R(\mathbf{r_i})$ and $H_{Q_i}(\mathbf{r_i})$ is measured as the scalar product between their normal components (see Section 3.2).

In practice, the points in $R$ are visited in random order. If $\mathbf{r_i}$ can be included in $P_0$, all points in the $h$-neighborhood around $\mathbf{r_i}$ are discarded for inclusion in $P_0$. The process terminates after all points are tested.

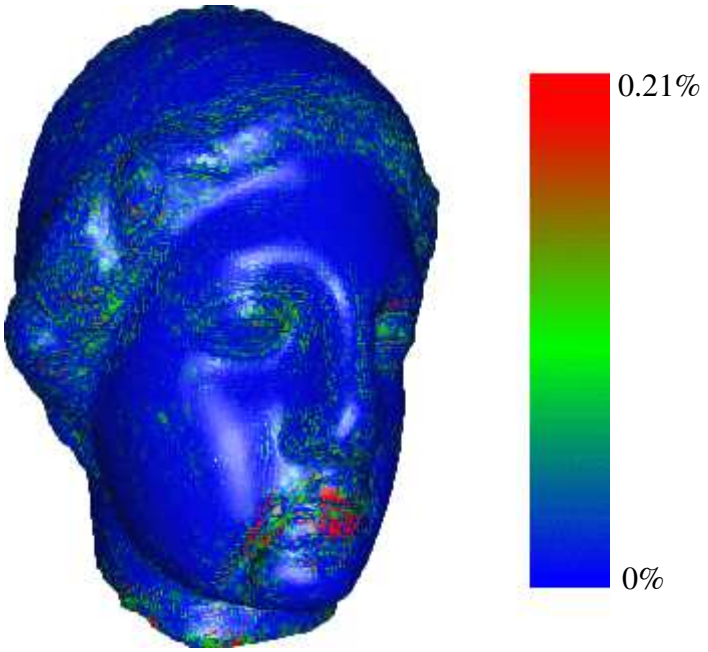



Figure 3.4: A color-coding of the magnitude of the displacement for the Venus model. Note that smooth regions have small displacements, while regions containing fine detail need larger displacements The magnitude of the displacements essentially corresponds to the energy in the respective frequency band.

## 3.4 The PPSS encoding

Figure 3.4 illustrates the magnitudes of the displacements of the progressive set. Observing that the vast majority of the displacements are of small magnitudes give rise to a space efficient encoding scheme. To create an encoding of $P_{i+1}$ given $P_i$, we perform the routine described in Section 3.3.1. New points are generated and projected both on $S_{P_i}$ and on $S_R$. The difference between the two projections is the *displacement* denoted by $\Delta$. Since the distance between the two surfaces is small, the displacements are merely the details of the surface and can be encoded in a small number of bits.

To decode $P_{i+1}$, a reverse procedure is applied. New points are generated as described in the encoding procedure. For each point $\mathbf{r}$, the reference plane and polynomial are computed using Eqns. (3.1),(3.2). Then the point is projected using a modified Eq. (3.3) as follows:

$$MLS(\mathbf{r}) = \mathbf{r} + (t + g(0,0) + \Delta)\mathbf{n}. \tag{3.6}$$

For efficient storage, we quantize the displacement values to a user-specified accuracy. An error bound defines the maximal tolerated error with respect to the diagonal of the

object's bounding box. The range of the displacements and the error bound defines the number of bits required to properly represent the displacements. Recall that the decoding procedure is highly dependent on performing the exact same procedure that the encoder performed. Quantizing the values and reconstructing them creates minor differences that may lead to somewhat different sets of points that are added to each level. If this occurs the encoder and decoder may no longer be synchronized. Therefore, in the encoding process the displacements are quantized to guarantee that the decoder generates the exact set of points that is encoded.

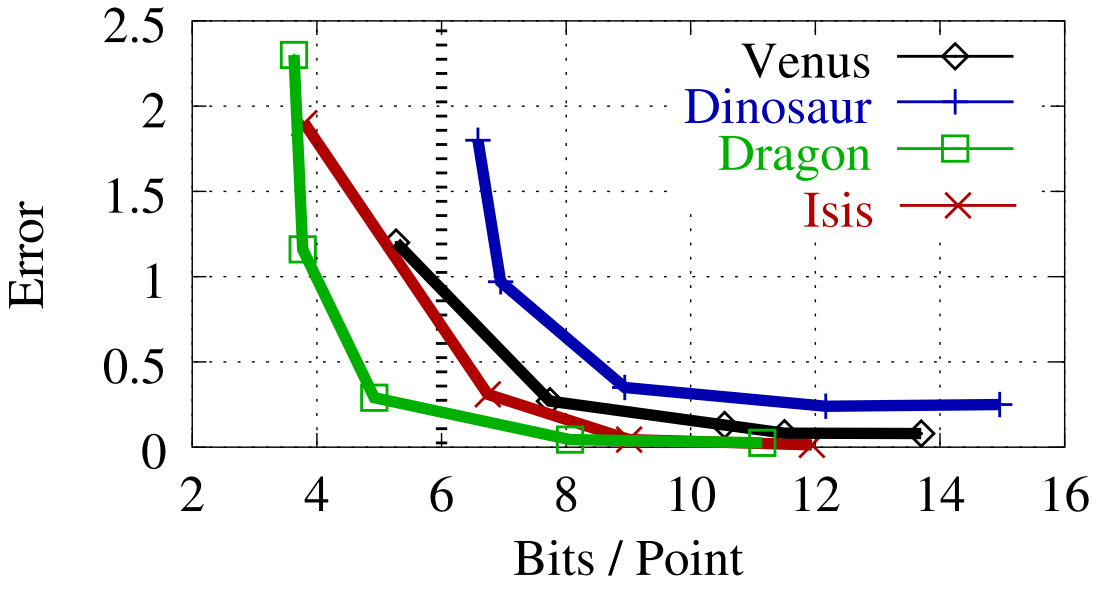



Figure 3.5: Rate distortion curve: shows a comparison of size vs. accuracy achievable with our compression method. The error is measured on a scale of $10^{-4}$ of the bounding box of the model.

## 3.5 Results

We have implemented the progressive point set representation as described in the previous sections and applied it to several models. Table 3.1 summarizes the results by showing the average number of bits / displacement required with respect to error tolerance. The error is expressed with respect to the diagonal of the bounding box of the given model. Note that the error is merely subject to the quantization applied to the displacements. For the models in Table 3.1, five (for the dino model) to seven (for the dragon model) levels were generated. Our experiments as shown in Table 3.2 and Figures 3.5 and 3.7 suggest that an average of

| Name | Points | Points in $P_0$ | bits / displacement (error $10^{-4}$) | | | |
|---|---|---|---|---|---|---|
| | | | **0.001** | **0.01** | **0.1** | **0.5** |
| Dragon | 437K | 16K | 9.7 (0.026) | 6.6 (0.044) | 3.4 (0.29) | 2.4 (1.6) |
| Isis | 187K | 9K | 9.9 (0.01) | 6.9 (0.04) | 4.7 (0.31) | 1.8 (1.9) |
| Venus | 134K | 7K | 9.2 (0.08) | 8.3 (0.13) | 5.5 (0.27) | 3.0 (1.2) |
| Dino | 56K | 4K | 10.9 (0.25) | 8.1 (0.24) | 4.9 (0.35) | 2.9 (0.97) |

Table 3.1: Achieved bit-rates for given error bounds. The user can specify an error bound on the displacement values. Depending on the error bound, a quantization scheme is chosen, which influences the number of bits necessary to encode the displacements. The small number of quantization levels typically results in a systematically smaller error as compared to the error bound (shown in parentheses).

| **Inserted points** | **Bits / $\Delta$** | **Total points** | **Total size** | **Average b/p** | **error ($10^{-4}$)** |
|---|---|---|---|---|---|
| 476370 | 2.3 | 493142 | 139K | 3.66 | 2.7 |
| 472330 | 2.4 | 489102 | 144K | 3.77 | 1.6 |
| 440907 | 3.5 | 457679 | 192K | 4.92 | 0.29 |
| 439652 | 6.6 | 456424 | 364K | 8.06 | 0.044 |
| 439840 | 9.7 | 456612 | 534K | 11.1 | 0.026 |

Table 3.2: A comparison of the size vs. accuracy for the Dragon model. Each line shows the size of the model as a function of the number of bits used for quantization of the displacement values. The base point set contains $16772$ points compressed to $37.4$ bits / point.

five bits / point yields a pleasing visual quality. The base point set $P_0$ is compressed by triangulating the points using the *ball-pivoting algorithm* (BPA) [Bernardini et al. 1999] and applying a mesh compression tool [Gotsman et al. 2001]. The time to compress the models in Table 3.1 range from several minutes to 120 minutes on Pentium™ III 1Ghz; our code was not optimized and computes the MLS reference plane using nonlinear optimization.

Table 3.2 shows the compression achieved by varying the number of bits for the displacement values. We measured the accuracy of the reconstructed model in the spirit of Metro [Ciampalini et al. 1997], i.e. by sampling distances in normal direction. To measure the distance between an MLS surface $S_1$ defined by a set of points $P_1$ and the reference MLS surface $S$ defined by $P$, we sample arbitrary points in the neighborhood of $S_1$, and use the MLS projection procedure to project each point on $S_1$ and on $S$. The average difference between the two projections is the error.

We compared the PPSS–based compression technique with techniques for multiresolution mesh compression. In particular, we have used Khodakovsky's mesh compression technique [Khodakovsky et al. 2000] to generate meshes with increasing accuracy with respect to a reference mesh. The vertices of the reference mesh were used to build a PPSS. Since we do not have connectivity information the BPA was used to generate meshes from the point sets. The resulting meshes have been compared to the reference using Metro. Figure 3.6 shows a visualization of the results. Note that the PPSS does not fit the piecewise linear geometry of the reference mesh (as the mesh compression technique) but the MLS surface defined by the vertices. This adds some bias to the resulting error for the PPSS.

Figure 3.7 shows a series of progressively refined point sets, where the shaded images are rendered by an upsampling procedure [Alexa et al. 2001], which requires no displacements. The rendering performs a local refinement of the surface around each point of the model. The images in the second column of Figure 3.8 are rendered with the above upsampling method and the images in the third column are rendered using the OpenGL™ glPointSize function. Since the MLS surface is continuous and smooth, the quality of the upsampled renderings is higher than a splat rendering.

The MLS surface is smooth and as such does not reconstruct sharp features (since it is not able to model discontinuities in its derivatives). While reconstructing point samples of CAD models with sharp features (see Figure 3.9, those sharp features are smoothed out, while the rest of the object is reconstructed faithfully. Our method deals with boundaries by computing the convex hull of the neighborhood of a point (as previously described in Section 3.3.1). For sharp edges in boundaries like the rectangular hole in Figure 3.9c and d, our method converges to round corners with radius of the size of the input feature size, that is, the spacing between points.

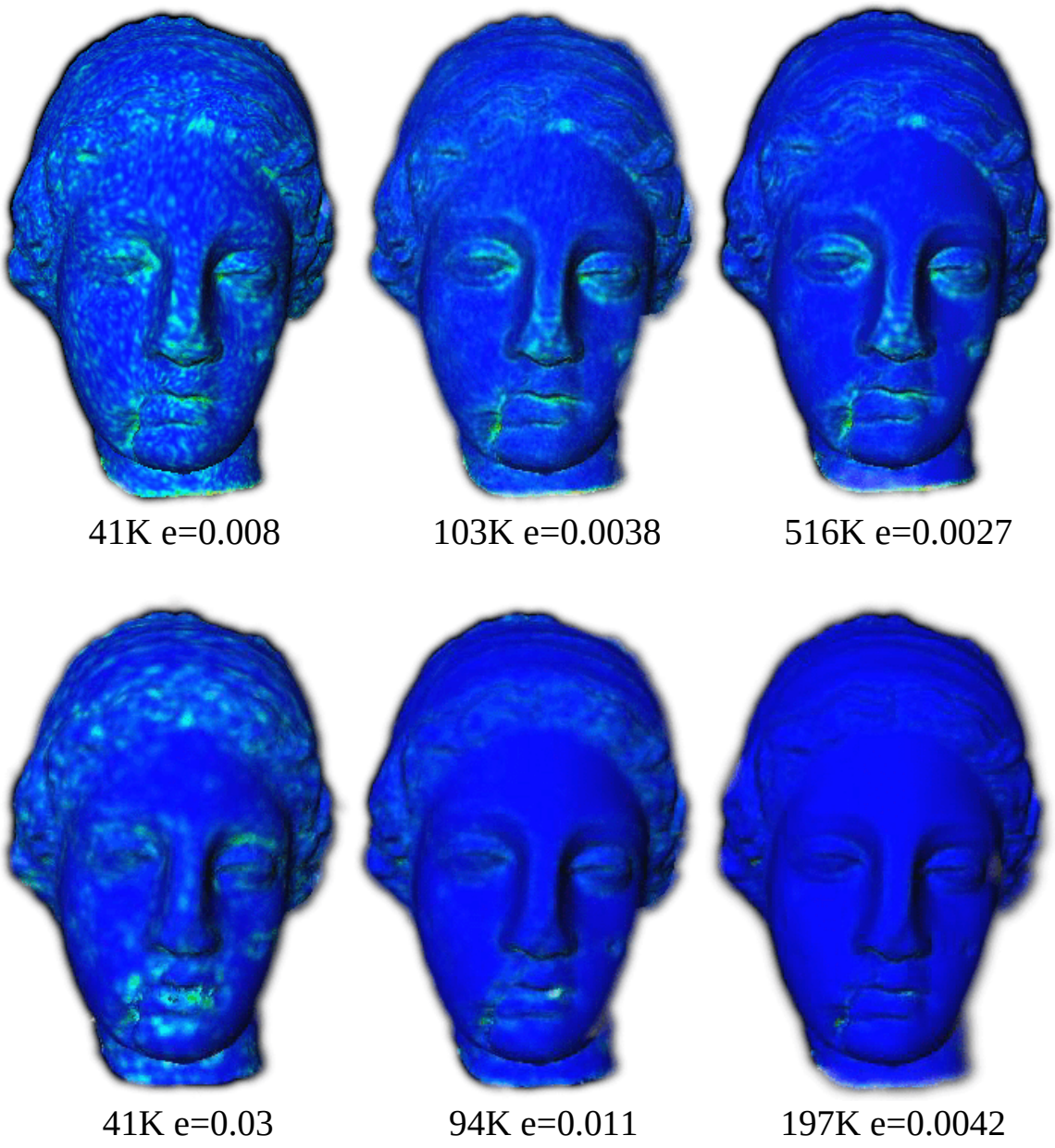



Figure 3.6: Comparison of meshes using Metro. The top row displays several steps during the refinement process of Khodakovsky's algorithm. The numbers below the figures show their size and mean error with respect of the bounding box of the object, as reported by the I.E.I-CNR Metro tool. The bottom row displays three meshes reconstructed from a PPSS and compared to the input mesh. Note that the PPSS does not fit the reference mesh but rather the smooth MLS-surface over the vertices. The point sets are triangulated using BPA to be able to apply Metro. Color ranges from blue to green with respect to the error. Note that the Metro tool normalizes the color map to the maximal error of the model being colored.

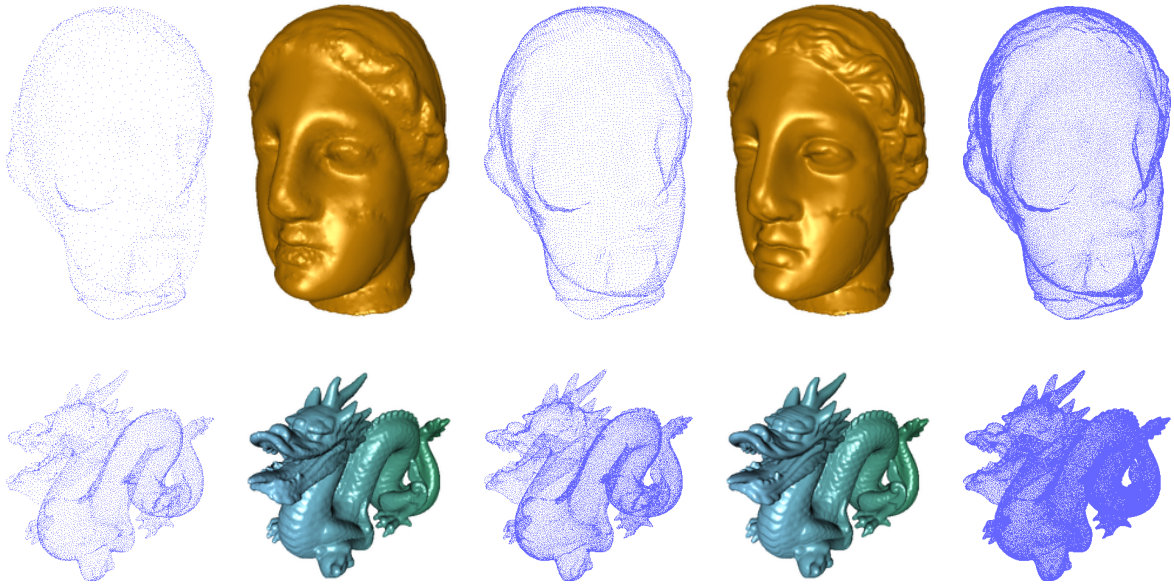


Figure 3.7: A progressive series of the Venus and Dragon models.

## 3.6 Discussion

As in other multilevel shape representations, the levels essentially correspond to different bands in the spectrum of the shape. The Gaussian weight function leads to a Gaussian filtering of the shape (compare Eqns. (3.1) and (3.2) ). The spatial radius $h$ of the filter is inversely related to the Gaussian in the frequency domain. Thus, the base point set represents the shape with the most relative energy in the low frequency bands, while the refinement levels add higher frequency bands (see the shape hierarchy in Figure 3.1, and the details in Figure 3.3). The projection operator allows us to compute the scalar displacement necessary to lift a point from one level to the next. The displacements are with respect to local frames (as in [Kobbelt et al. 2000]). The magnitude of this displacement is, thus, a measure of the energy in the respective frequency band. This is illustrated with color codes in Figure 3.4.

The MLS surface definition is based on differential geometry, namely, that the surface can be locally approximated by a function. If this assumption is not met, we fail to define the plane (Eq.3.1) and cannot reconstruct the surface. This ill condition happens when the surface is not sampled densely enough in areas of high curvature or in areas with low SNR, i.e., when the noise is larger than the expected feature size. These conditions can be identified, see [Alexa et al. 2003] for more details.

Because point-sampled objects contain no explicit topological information, it is necessary to make assumptions about the underlying sampling density in order to properly

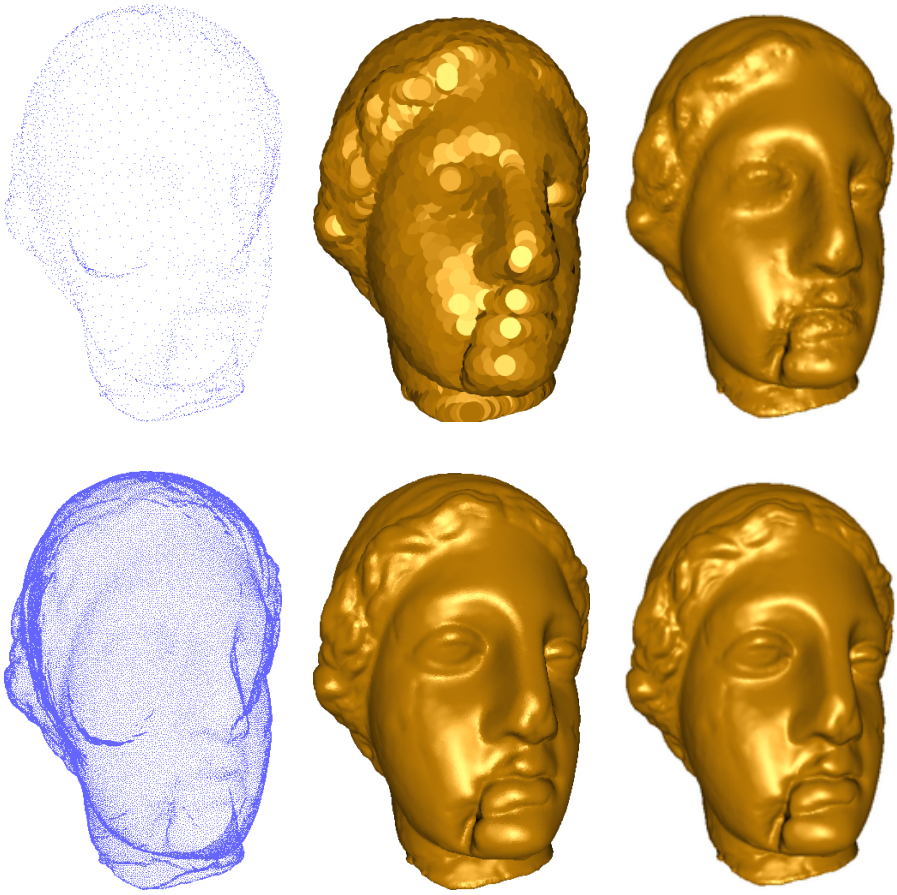
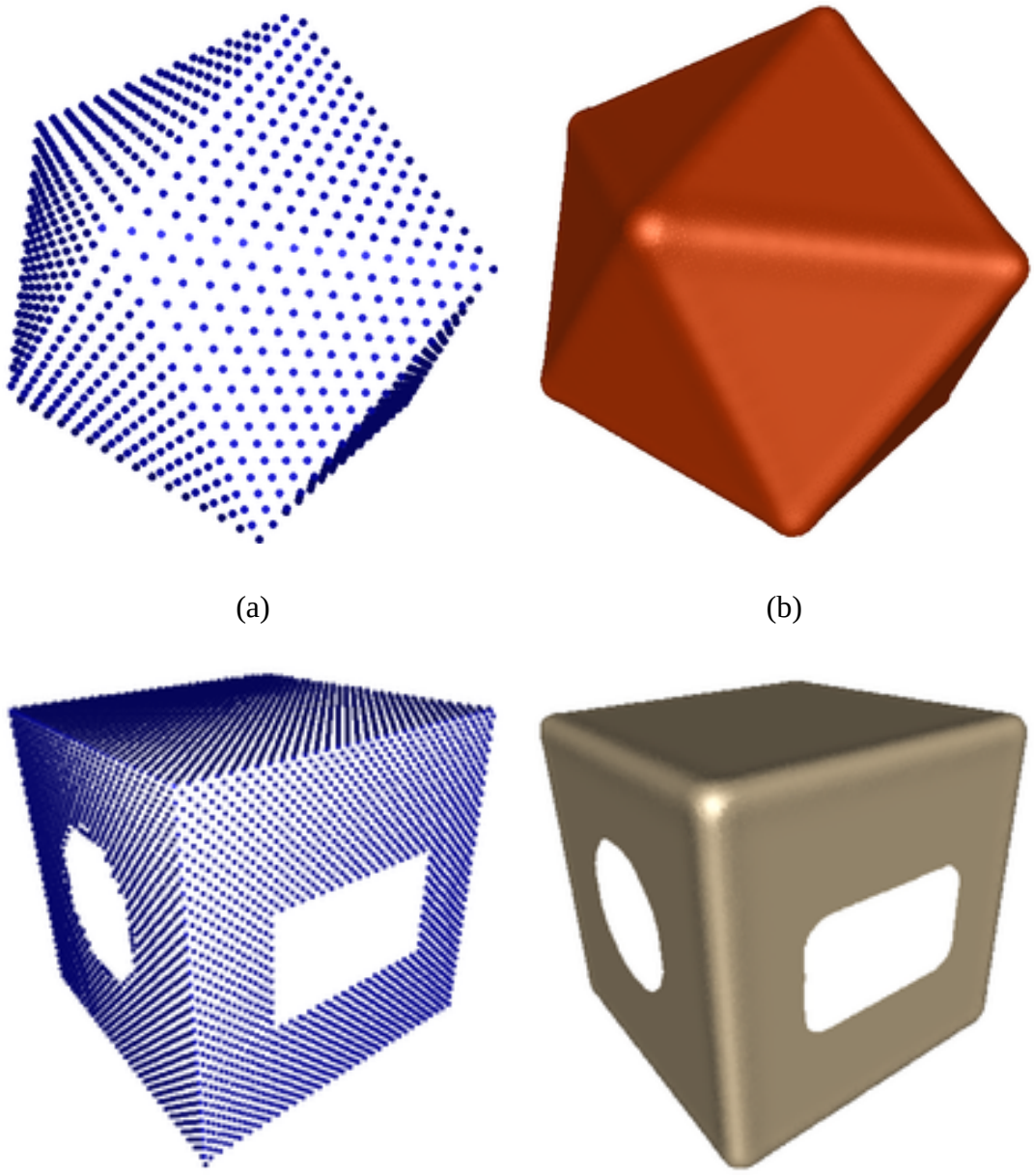


Figure 3.8: Two intermediate representations of the Venus model in the hierarchy. On the left we show the set of points. In the middle, the set of points are rendered by splatting using OpenGL™. The images on the right are rendered using an MLS upsampling procedure, requiring no additional data.



(a) (b)

(c) (d)

Figure 3.9: Reconstruction of sharp edges and boundaries: On the left (a) and (c), we show the input models. (b) and (d) are the reconstructions of the models using our algorithm.

reconstruct this connectivity information. In general, this is a tough problem, and substantial practical and theoretical work has been performed in the area [Amenta and Bern 1999]. Provably good techniques, e.g. [Amenta et al. 1998], have been shown to be effective, but quite slow for actual use, mostly due to the need to compute 3D Delaunay triangulation of the complete point sets. For modeling point sets, simpler (and computationally cheaper) techniques have been used. Zwicker et al. [2002] and Pauly et al. [2002] use $k$-nearest neighbor queries to reconstruct a neighborhood. This works well for objects with fairly isotropic and uniform sampling density; furthermore the $k$-nearest neighbor query may unintuitively fill "real" holes in the model. Linsen [2001] used the *angle criterion* for choosing the neighbors of a points, assuming the object is of genus zero, or otherwise defined some threshold on the point spacing. The approach we currently use for defining the neighborhood of a point is to query for nearest neighbors in a user defined radius, assuming that the distribution of points is fairly uniform. This assumption is reasonable for scanned objects. As shown in [Alexa et al. 2003], this method can deal with some variation in the input point density, but will fail if the density varies significantly.

Recently, Kalaiah and Varshney [2001] introduced an effective method for rendering point primitives that requires the computation of the principal curvatures and a local coordinate frame for each point. This approach is natural for the MLS surface representation since it requires a local coordinate frame and the principal curvature for each. During the MLS projection procedure a local coordinate frame is computed, and the principal curvatures can be estimated analytically from Eq. (3.2).

# Chapter 4

# Bilateral Mesh Denoising

With the proliferation of 3D scanning tools, interest in removing noise from meshes has increased. The acquired raw data of the sampled model contains additive noise from various sources. It remains a challenge to remove the noise while preserving the underlying sampled surface, in particular its fine features. Related techniques like mesh smoothing or mesh fairing alter the given surface to increase its degree of smoothness. The goal of these techniques is to create semi-uniform triangulation, often with subdivision connectivity. This work focuses on mesh denoising, which is an important preprocess for many digital geometry applications that rely on local differential properties of the surface.

Denoising the sampled data can be applied either before or after generating the mesh. The advantage of denoising a mesh rather than a point-cloud, is that the connectivity information implicitly defines the surface topology and serves as a means for fast access to neighboring samples. The information in a mesh can be separated into two orthogonal components: a *tangential* and a *normal* component. The normal component encodes the geometric information of the shape, and the tangential component holds parametric information [Guskov et al. 1999]. In this formulation, moving vertices along their normal directions, modifies only the geometry of the mesh. Related to this notion are evolution curves [Osher and Sethian 1988], where points are shifted in the direction of the normal at a distance that is proportional to their curvature, to get smoother curves over time. Our denoising method is based on this idea, shifting mesh vertices along their normal direction.

The extensive research on image denoising serves as a foundation for surface denoising and smoothing algorithms. However, adapting these algorithms from the two dimensional plane to a surface in three dimensions is not straightforward for three main reasons: (i) **Irregularity**; unlike images, meshes are irregular both in connectivity and sampling, (ii) **Shrinkage**; image denoising algorithms are typically not energy preserving. While this is less noticeable in images, in meshes, this is manifested as shrinkage of the mesh, (iii) **Drifting**; naive adaptation of an image denoising technique may cause artifacts known as *vertex drifts*, in which the regularity of the mesh decreases.

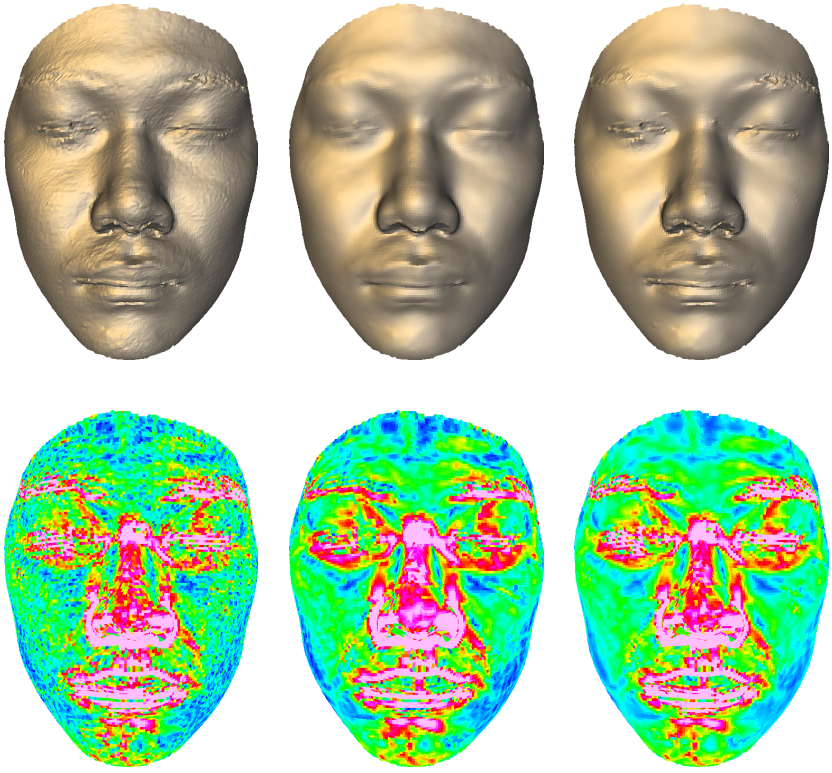


Figure 4.1: Denoising a scanned model: On the left is the input model, in the middle is the result of implicit fairing [Desbrun et al. 1999], and on the right is the result of our algorithm. The top row visualizes the details of the models, and on the bottom row is a mean curvature visualization. Data courtesy of Alexander Belyaev.

The bilateral filter, introduced by Tomasi and Manduchi [1998], is a nonlinear filter derived from Gaussian blur, with a feature preservation term that decreases the weight of pixels as a function of intensity difference. It was shown that bilateral filtering is linked to anisotropic diffusion [Barash 2002], robust statistics [Durand and Dorsey 2002], and Bayesian approaches [Elad 2001]. Despite its simplicity, it successfully competes with image denoising algorithms in the above categories. The bilateral filtering of images and its adaptation to meshes has an intuitive formulation, which leads to a simple method for selecting the parameters of the algorithm.

The contribution of this chapter is a mesh denoising algorithm that operates on the geometric component of the mesh. The origin of the denoising algorithm is the bilateral filter that has a simple and intuitive formulation, is fast and easy to implement, and adapting it to meshes, yields results that are as successful as its predecessor.

## 4.1 Previous work

Image denoising is part of on-going research in image processing and computer vision. The state-of-the-art approaches to image denoising include: wavelet denoising [Donoho 1995], nonlinear PDE based methods including total-variation [Rudin et al. 1992], and bilateral filtering [Tomasi and Manduchi 1998]. These approaches can be viewed in the framework of basis pursuit [Chen et al. 1999].

Typically, mesh denoising methods are based on image denoising approaches. Taubin [1995] introduced signal processing on surfaces that is based on the definition of the Laplacian operator on meshes. Peng et al. [2001] apply locally adaptive Wiener filtering to meshes. Geometric diffusion algorithms for meshes was introduced by Desbrun et al. [1999], they observed that fairing surfaces can be performed in the normal direction. Anisotropic diffusion for height fields was introduced by Desbrun et al. [2000], and Clarenz et al. [2000] formulated and discretized anisotropic diffusion for meshes. Recently, Bajaj and Xu [2003] achieved impressive results by combining the limit function of Loop subdivision scheme with anisotropic diffusion. Tasdizen et al. [2002] apply anisotropic diffusion to normals of the level-set representation of the surface, and in the final step, the level-set is converted to a mesh representation. Guskov et al. [1999] introduced a general signal processing framework that is based on subdivision, for which denoising is one application.

## 4.2 Bilateral mesh denoising

We open with a description of our method for filtering a mesh using local neighborhoods. The main idea is to define a local parameter space for every vertex using the tangent plane to the mesh at a vertex. The heights of vertices over the tangent plane are synonymous with the gray-level values of an image, and the closeness components of the filter are the tangential components. The term *offset* is used for the heights. Let $S$ denote the noise-free surface, and let M be the input mesh with vertices that sample $S$ with some additive noise. Let $\mathbf{v} \in M$ be a vertex of the input mesh, $d_0$ its signed-distance to $S$, and $\mathbf{n_0}$ the normal to $S$ at the closest point to $\mathbf{v}$. The noise-free surface $S$ is unknown and so is $d_0$, therefore we estimate the normal to the surface as the normal $\mathbf{n}$ to the mesh, and $d$ estimates $d_0$ as the application of the filter, updating $\mathbf{v}$ as follows:

$$\hat{\mathbf{v}} = \mathbf{v} + d \cdot \mathbf{n}. \tag{4.1}$$

Note that we do not have to define a coordinate system for the tangential component; as explained below, we apply a one-dimensional filter with a spatial distance as a parameter. The filter is applied to one vertex at a time, computing a displacement for the vertex and updating its position.

### 4.2.1 Bilateral filtering of images.

Following the formulation of Tomasi and Manduchi [1998], the bilateral filtering for image $I(\mathbf{u})$, at coordinate $\mathbf{u} = (x, y)$, is defined as:

$$\hat{I}(\mathbf{u}) = \frac{\sum_{\mathbf{p} \in N(\mathbf{u})} W_c(\|\mathbf{p} - \mathbf{u}\|) W_s(|I(\mathbf{u}) - I(\mathbf{p})|) I(\mathbf{p})}{\sum_{\mathbf{p} \in N(\mathbf{u})} W_c(\|\mathbf{p} - \mathbf{u}\|) W_s(|I(\mathbf{u}) - I(\mathbf{p})|)}, \tag{4.2}$$

where $N(\mathbf{u})$ is the neighborhood of $\mathbf{u}$. The closeness smoothing filter is the standard Gaussian filter with parameter $\sigma_c$: $W_c(x) = e^{-x^2/2\sigma_c^2}$, and a feature-preserving weight function, which we refer to as a *similarity weight function*, with parameter $\sigma_s$ that penalizes large variation in intensity, is: $W_s(x) = e^{-x^2/2\sigma_s^2}$. In practice, $N(\mathbf{u})$ is defined by the set of points $\{\mathbf{q_i}\}$, where $\|\mathbf{u} - \mathbf{q_i}\| < \rho = \lceil 2\sigma_c \rceil$.

### 4.2.2 Algorithm

We begin introducing the algorithm by describing how to compute the normal and tangential components that are assigned to Eq. 4.2, yielding a new offset $d$. Since $S$ is unknown, and we wish to use the edge preservation property of the bilateral filter, we define $S_v \subset S$ as the smooth connected component of $S$ that is closest to $\mathbf{v}$. For the normal component, we would like to compute the offsets of the vertices in the neighborhood of $\mathbf{v}$, denoted by $\{\mathbf{q_i}\}$, over the noise-free smooth component $S_v$. We use the tangent plane to $\mathbf{v}$ defined by the pair $(\mathbf{v}, \mathbf{n})$ as a first-order approximation to $S_v$. The offset of a neighbor $q_i$ is the distance between $q_i$ and the plane. The following is the pseudo-code for applying a bilateral filter to a single vertex:

```
DenoisePoint(Vertex v, Normal n)
  {q_i} = neighborhood(v)
  K = |{q_i}|
  sum = 0
  normalizer = 0
  for i := 1 to K
    t = ||v − q_i||
    h = ⟨n, v − q_i⟩
    w_c = exp(−t^2/(2σ_c^2))
    w_s = exp(−h^2/(2σ_s^2))
    sum += (w_c · w_s) · h
    normalizer += w_c · w_s
  end
  return Vertex v̂ = v + n · (sum/normalizer)
```

The plane that approximates the noise-free surface should on one hand, be a good approximation of the local surface, and on the other hand, preserve sharp features. The first requirement leads to smoothing the surface, while the latter maintains the noisy surface. Therefore, we compute the normal at a vertex as the weighted average (by the area of the triangles) of the normals to the triangles in the 1-ring neighborhood of the vertex. The limited neighborhood average smoothes the local surface without over-smoothing. In some cases, for example, of a synthetic surface, the normal of an edge vertex will erroneously point to the average direction and lead to a rounded edge.

For the tangential component, the correct measure of distance between vertices on the smooth surface is the geodesic distance between points. Since we use local neighborhoods, we approximate the geodesic distance using the Euclidean distance. This approximation seems to introduce artifacts in the neighborhood of sharp features, since vertices that happen to be geodesically far from the smoothed vertex may be geometrically close. Furthermore, the assumption from differential geometry that a neighborhood of a point on a surface can be evaluated by a function over the tangent plane to that point may not be satisfied. Both apparent problems do not hinder our algorithm because any of the above offending vertices is penalized by the similarity weight function.

### 4.2.3 Mesh shrinkage and vertex-drift

Image denoising and smoothing algorithms that are based on (possibly weighted) averaging of neighborhood, result is shrinkage of the object. Taubin [1995] solves this problem for the Laplacian operator by alternating shrink and expand operations. Another common

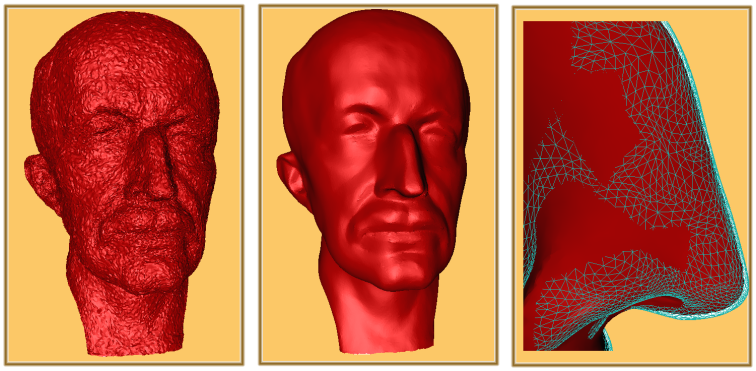


Figure 4.2: The shrinkage problem. On the left the model of Max Planck with heavily added random noise. In the middle the denoised model after four iterations of our algorithm without volume preservation. The Max-Planck model is courtesy of Christian Rössl from Max Planck Insitut für Informatik.

approach is to preserve the volume of the object as suggested by Desbrun et al. [Desbrun et al. 1999].

Our algorithm, also shrinks the object. This can be observed when smoothing a vertex that is a part of a curved patch; the offset of the vertex approaches the average of the offsets in its neighborhood. Therefore, we follow the volume preservation technique.

Vertex-drift is caused by algorithms that change the position of the vertices along the tangent plane as well as the normal direction. The result is an increase in the irregularity of the mesh. Our algorithm moves vertices along the normal direction, and so, no vertex-drift occurs.

### 4.2.4 Handling boundaries

Often meshes, in particular scanned data sets, are not closed. There are two aspects to note here: first, the shape of the boundary curve, which is the related problem of “mesh fairing”. Second, is that a filter is defined for a neighborhood of a point. However for boundary points, part of the neighborhood is not defined. One common solution to this problem is to define a virtual neighborhood by reflecting vertices over edges. Our filter inherently handles boundaries by treating them as sharp edges with virtual vertices at infinity. The similarity weight sets the weight of virtual vertices to zero, and thus, the normalization of the entire filter causes boundaries to be handled correctly.

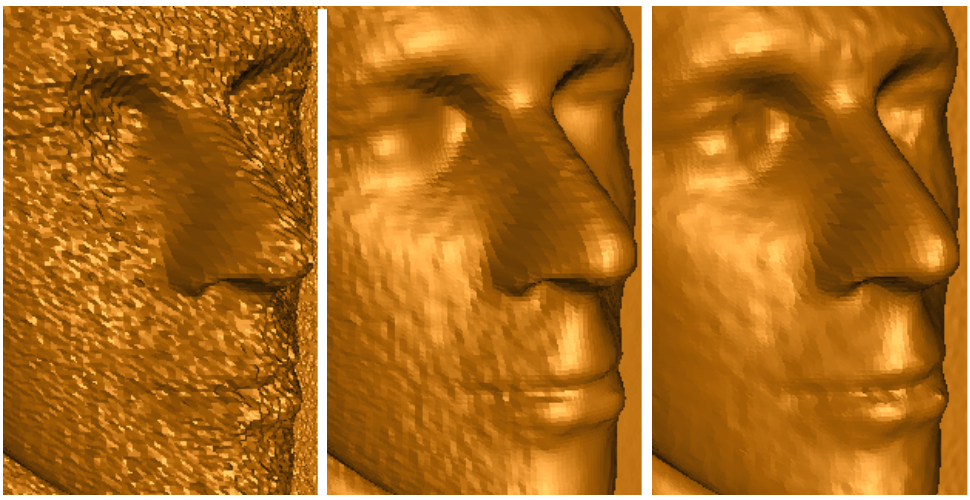


Figure 4.3: Comparison with AFP. On the left is the input model, in the middle is the result denoised by AFP, and on the right is the result of bilateral mesh denoising. Observe the difference in details in the area of the lips and eyes. The noisy model and the AFP denoised models are courtesy of Mathieu Desbrun.

### 4.2.5 Parameters

The parameters of the algorithm are: $\sigma_c$, $\sigma_s$, the kernel size $\rho$, and the number of iterations. We propose an intuitive user-assisted method for setting these parameters. Two parameters, $\sigma_c$ and $\sigma_s$ are interactively assigned: the user selects a point of the mesh where the surface is expected to be smooth, and then a radius of the neighborhood of the point is defined. The radius of the neighborhood is assigned to $\sigma_c$, and we set $\rho = 2\sigma_c$. Then $\sigma_s$ is set to the standard deviation of the offsets in the selected neighborhood.

One may choose a large $\sigma_c$ and perform a few iterations, or choose a narrow filter and increase the number of iterations. Multiple iterations with a narrow filter has the effect of applying a wider filter, and results in efficient computation. Using a small value for $\sigma_c$ is sensible for two reasons: (i) large values may cross features as shown in, and (ii) smaller values result in a smaller neighborhood which leads to faster computation of every iteration.

In all the results shown in this paper, we used up to five iterations, we found a small number of iterations sufficient for our purposes, and advantageous both to the speed of computation and for the numerical stability of the filter.

Noisy data may lead to unstable computation of the normals if the 1-ring neighborhood of a vertex is used to compute the normals. For extremely noisy data, the normal to a vertex is computed using the k-ring neighborhood of the vertex, where $k$ is defined by the user. For every scanned models that we denoised, the normals were computed using the 1-ring neighborhoods. Note that only for the Max Palanck (Figure 4.2) model, we were required

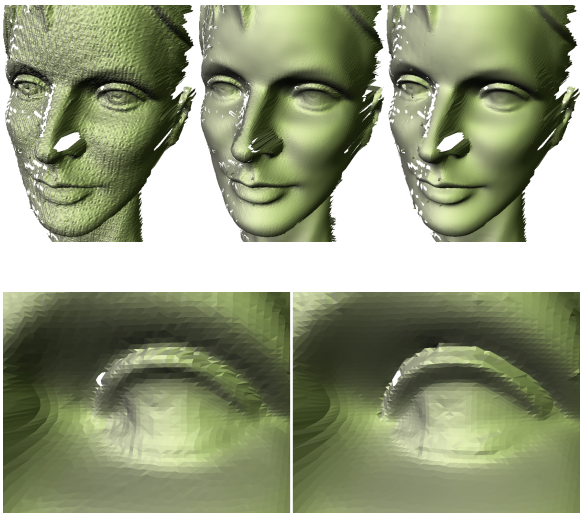


Figure 4.4: Results of denoising the face model. On the top row from left to right are the input noisy mode, the results of [Jones et al. 2003], and our result. On the bottom we zoom on the right eye of the model, where the bottom left image shows the results of Jones et al. , and on the bottom right is the result of our method. The face model is courtesy of Jean-Yves Bouguet.

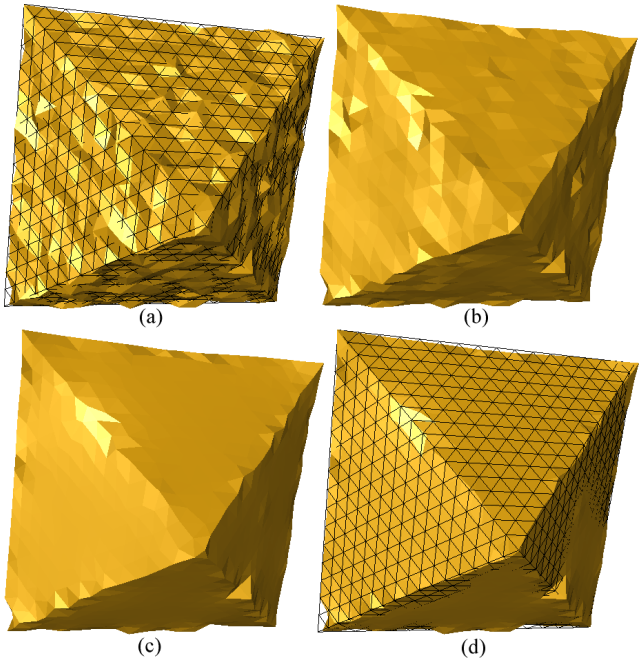



Figure 4.5: Denoising of CAD-like model. (a) is the input noisy model, (b) is the result of two iterations of our algorithm, (c) and (d) are the result of five iterations of our algorithm, where in (d) the clean model was superimposed on the denoised model.

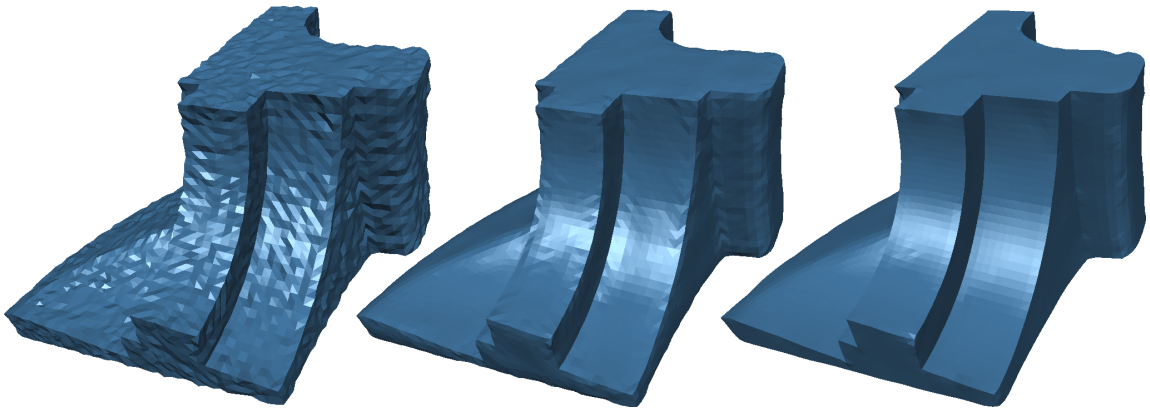


Figure 4.6: Results of denoising the Fandisk model. On the left is the input noisy model, in the middle is the results of [Jones et al. 2003], and on the right is our result.

to use the 2-ring neighborhood to compute normals.

## 4.3 Results

We have implemented the bilateral mesh denoising algorithm as described in the previous section and compared our results to the results of the anisotropic denoising of height fields algorithm (*AFP*) [Desbrun et al. 2000], Jones et al. [2003], and the *implicit fairing (IF)* algorithm [Desbrun et al. 1999] [1]. The times are reported on a 1GHz Pentium™ III. In Figure 4.3, we compare the AFP algorithm with our results. The mesh with 175K vertices is smoothed by three iterations in 10 seconds. Observe the preserved details near the mouth of the model denoised by our algorithm. Figure 4.1 shows a comparison with implicit fairing. The smoothness of the object can be appreciated from the visualization of the mean curvature in the second row. The model has 17K vertices, and it was denoised in three iterations, taking 1.8 seconds. In Figure 4.5 we show the denoising of a CAD object. Observe that the sharp features are preserved, but vertices with larger error are treated as outliers and thus are not smoothed out. For the Max Planck model (Figure4.2) (100k vertices), the timing for a single iteration is 5.75 seconds, and the total number of iterations was four.

Independently, Jones et al. [2003] present a similar algorithm that uses bilateral filtering for smoothing surfaces. The main difference between the two methods is in the surface predictor. More specifically, Jones et al. take the distance between the point and its projection onto the plane of a neighboring triangle, whereas our approach takes the distance between a neighboring point and the tangent plane. While we perform several filtering iterations,

[1] Implementation courtesy of Y. Ohtake

Jones et al. smooth a surface in a single pass. Figure 4.4 and 4.6 compare the results for two different types of models.

Volume preservation is a global operation, whereas denoising is a local operation. In Figure 4.2, we visualize the local change of volume by adding severe synthetic noise (of zero mean and variance of 0.05% of the diagonal of the bounding box of the model) to the clean model of Max Planck, and then denoised the model. On the right, we zoom on a feature of the model, superimposing the original mesh on top of the smoothed model, showing that the change in shape and volume is minimal.

## 4.4 Summary

We presented a mesh-denoising algorithm that modifies vertices in the normal direction. The bilateral filtering algorithm that we use is practical, clear and simple. The proposed method deals with irregular meshes and does not perform any reparameterization. In addition, the only property of the mesh that we use is the topological information, and therefore, the algorithm can be adapted to point-based representations.

# Chapter 5

# Conclusions and future work

This work describes a point-based surface representation that is smooth, local and resilient to sampling error (noise). The surface definition combines the usability of point-based modeling with a consistent surface definition. The heart of the surface definition is a projection operator from a point near the surface to the surface.

In this work, we focused on the introduction of the projection operator and computing it. We have also used it for surface resampling and rendering, as well as a progressive surface representation and noise removal. These are only a small number of applications for the presented surface definition. We believe that it will foster future research in point-based surface representation and applications; a number of researchers have already used our surface definition [Adamson and Alexa 2003a; Adamson and Alexa 2003b; Igarashi and Hughes 2003; Mederos et al. 2003; Pauly et al. 2002; Pauly et al. 2003; Wang and Oliveira 2003].

## 5.1 Point set surfaces

In differential geometry, a smooth surface is characterized by the existence of smooth local maps at any point. In this work we use this as a framework to approximate a smooth surface defined by a set of points and we introduced new techniques to resample the surface to generate an adequate representation of the surface.

To render such surfaces, the surface is covered by a finite number, as small as possible, of non-conforming, overlapping, polynomial patches. We showed that the error of these approximations is bounded and dependent on the spacing among points. Thus, it is possible to provide a point set representation that conforms with a specified tolerance.

Our paradigm for representing surfaces advocates the use of a point set (without connectivity) as a representation of shapes. This representation is universal in the sense that it

is used from the beginning (i.e. acquisition) to the end (i.e. rendering) of a graphical representation's life cycle. Moreover, we believe that this work is a step towards rendering with higher order polynomials. Note that we have used simple primitives (points) to achieve this goal. This admits to the current trend of integrating high quality rendering features into graphics hardware.

It would be interesting to integrate our approach with combinatorial methods such as the one of Amenta et al. [1998]. This would combine topological guarantees with the additional precision of higher order approximations and the possibility of smoothing out noise or small features.

Using different values for $h$, it is easy to generate more smooth or more detailed versions of a surface from one point set (see, for example, Figure 2.4). A set of different versions could be used as a smooth-to-detailed hierarchy and would allow for multiresolution modeling [Kobbelt et al. 2000]. Of course, $h$ is not necessarily a global parameter and could be adapted to the local feature size. Varying $h$ has several implications and utility in handling point sets (see [Lee 2000] for a nice introduction to the issues in two dimensions), such as properly accounting for differences in sampling rate and levels of noise during the acquisition process. Also, non radial functions might be necessary to properly account for sharp features in the models.

## 5.2 Progressive point set surfaces

Our technique has several unique features. First of all, it works directly on points, avoiding the need to triangulate the source point set, and the costly remeshing of the surface into subdivision connectivity. This is especially important for large and detailed datasets. Another important property of our technique is its locality. This leads to a numerically stable computation, linear time and space complexity and small memory footprint, which may lead to the development of out of core algorithms. Furthermore, progressive point set representations lead to a compression scheme, where the range of the error decreases with each level in the hierarchy. As shown above, at each level it is possible to upsample the surface from the sparse representation.

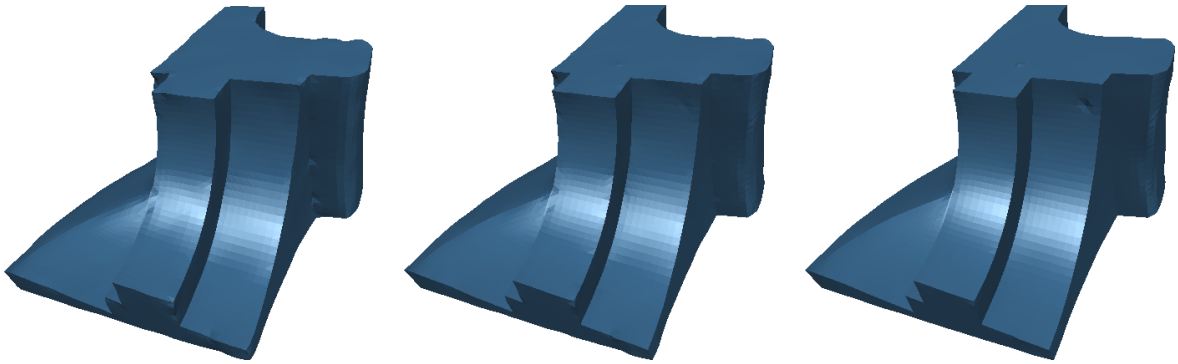
The lack of connectivity in the representation could also be the source of shortcomings. As shown in [Amenta et al. 1998], there are limits on surface samplings which can be proven to define a given surface. That is, our approach might need a relatively dense base set to resolve possible ambiguities in the modelling of certain complex surfaces. Moreover, it is considerably easier to handle discontinuities by triangulated models.

A considerable amount of research has been devoted to developing and optimizing the "mesh-based world". Many advanced methods require a local parameterization and local differential properties. The MLS projection procedure inherently has these qualities; for

example, texture synthesis techniques may be applied on surfaces using surface marching similar to [Turk 1992; Turk 2001]. We believe that the importance of point set representation and of MLS surfaces in particular is likely to play an increasing role in 3D geometric modelling.

## 5.3 Bilateral mesh denoising

Choosing the tangent plane as an approximation of $S_v$ is an important component of our algorithm. Positioning the plane at the average of the offsets of neighboring points would improve our approximation of the smooth surface. However, this would nullify the effect of the similarity term of the bilateral filter. We expect that computing the offsets over a higher order function such as a polynomial of low degree will reduce the shrinkage problem. Finding a high-order, feature-preserving function for the noise-free surface is a difficult problem. In the future, we would like to investigate this possibility, by combining the bilateral filter with a rational function.



(a) (b) (c)

Figure 5.1: Denoising with discontinuities in normals: (a) the result that was presented in Chapter 4; (b) the result of clustering normals, observe that the sharp edge with varying angle on the left of the figure is smoothed out toward the bottom; (c) the result of denoising using representative normals in all directions.

The key points of our algorithm are the choice of tangent plane and moving vertices along the normal direction. However this change in vertex position may lead to self-intersection of the denoised mesh. During the application of our algorithm to vertices on two sides of the edge, each vertex moves inwards; for sharp edges this will cause self-intersection after a number of iterations that is proportional to the angle of the edge.

The algorithm that we present assumes that the mesh is sampled regularly. This assumption is made when we fix the values of $\sigma_c$. Highly irregular meshes are uncommon in

scanned data-sets. To handle irregular data-sets, the parameters must be adjusted locally.

We use a single normal to a vertex, however, for edges there are two normals, one from the “left” and one from the “right”; a corner of a cube has three normals etc. It is possible to cluster normals pointing in a similar direction, trying to identify edges and corners, however this solution requires threshold, which is always hard to set, as can be seen in Figure 5.1 b. In preliminary experiments we have made, we used normals in all directions, or put in other words set the threshold to zero. Using this last idea, we get promising results as can be seen in Figure 5.1 c.

# Acknowledgements

We would like to thank the many people who have made their models avaialble. The Isis, Igea and Dinosaur are courtesy of Cyberware, the Budha, Bunny and Dragon models are courtesy of the Stanford 3D scanning repository , The scanned face is courtesy of Alexander Belyaev, The Max Planck-Institut für Informatik for the model of Max Planck, Mathieu Desbrun for the noisy face model, Jean-Yves Bouguet for the female face model, and Peter Neugebauer at Fraunhofer IGD in Darmstadt for the angel model.

This work would not be possible without the many people who made their software available: Fausto Bernardini for his implementation of the Ball-Pivoting Algorithm; Igor Guskov, Andrei Khodakovsky, Peter Schroeder, and Wim Sweldens for their remeshing and compression code; Craig Gotsman and Virtue Ltd for “Optimizer”, The Visual Computing Group of CNR-Pisa for Metro, Y. Ohtake for his implementation of “MeshEditor”.

# References

ADAMS, B., AND DUTR?, P. 2003. Interactive boolean operations on surfel-bounded solids. *ACM Transactions on Graphics (TOG) 22*, 3, 651–656.

ADAMSON, A., AND ALEXA, M. 2003. Approximating and intersecting surfaces from points. In *Eurographics Symposium on Geometry Processing*.

ADAMSON, A., AND ALEXA, M. 2003. Ray tracing point set surfaces. In *Shape Modeling International*.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2001. Point set surfaces. In *IEEE Visualization 2001*, 21–28. ISBN 0-7803-7200-x.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (January), 3–15.

AMENTA, N., AND BERN, M. 1999. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry 22*, 481–504.

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new voronoi-based surface reconstruction algorithm. *Proceedings of SIGGRAPH 98* (July), 415–422. ISBN 0-89791-999-8. Held in Orlando, Florida.

ARAVIND, AND VARSHNEY, A. 2003. Modeling and rendering of points with local geometry. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (January), 30–42.

BAJAJ, C. L., AND XU, G. 2003. Anisotropic diffusion of subdivision surfaces and functions on surfaces. *ACM Transactions on Graphics (TOG) 22*, 1, 4–32.

BAJAJ, C. L., BERNARDINI, F., AND XU, G. 1995. Automatic reconstruction of surfaces and scalar fields from 3D scans. *Computer Graphics 29*, Annual Conference Series (Nov.), 109–118.

BARASH, D. 2002. A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 6.

BARNES, J. E., AND HUT, P. 1986. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature 324*, 6270, 446–449.

BEATSON, R. K., AND NEWSAM, G. N. 1992. Fast evaluation of radial basis functions. *Computers and Mathematics with Applications 24*, 12, 7–19.

BERNARDINI, F., MITTLEMAN, J., RUSHMEIER, H., SILVA, C., AND TAUBIN, G. 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (October - December), 349–359. ISSN 1077-2626.

BESL, P. J., AND MCKAY, N. D. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and machine Intelligence 14*, 2 (Feb.), 239–258.

BÉZIER, P. E. 1968. How renault uses numerical control for car body design and tooling. In *Society of Automotive Engineer's Congress*. Detroit, MI.

BLOOMENTHAL, J., BAJAJ, C., BLINN, J., CANI-GASCUEL, M.-P., ROCKWOOD, A., WYVILL, B., AND WYVILL, G. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers, Inc.

BLOOMENTHAL, J. 1988. Polygonization of implicit surfaces. *Computer Aided Geometric Design 5*, 4, 341–355.

BLOOMENTHAL, J. 1994. An implicit surface polygonizer. In *Graphics Gems IV*, P. Heckbert, Ed. Academic Press, Boston, 324–349.

BOISSONNAT, J.-D. 1984. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics 3*, 4 (Oct.), 266–286.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3d objects with radial basis functions. *Proceedings of SIGGRAPH 2001* (August), 67–76. ISBN 1-58113-292-1.

CATMULL, E., AND CLARK, J. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10* (Sept.), 350–355.

CHAI, J.-X., CHAN, S.-C., SHUM, H.-Y., AND TONG, X. 2000. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 307–318.

CHEN, B., AND NGUYEN, X. 2001. POP: A hybrid point and polygon rendering system for large data. In *Proceedings Visualization 2001*, T. Ertl, K. Joy, and A. Varshney, Eds., IEEE Computer Society Technical Committee on Visualization and Graphics Executive Committee, 45–52.

CHEN, S. E., AND WILLIAMS, L. 1993. View interpolation for image synthesis. *Computer Graphics 27*, Annual Conference Series, 279–288.

CHEN, S. S., DONOHO, D. L., AND SAUNDERS, M. A. 1999. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing 20*, 1, 33–61.

CIAMPALINI, A., CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 1997. Multiresolution decimation based on global error. *The Visual Computer 13*, 5, 228–246. ISSN 0178-2789.

CIGNONI, P., FLORIANI, L. D., MONTONI, C., PUPPO, E., AND SCOPIGNO, R. 1994. Multiresolution modeling and visualization of volume data based on simplicial complexes. *1994 Symposium on Volume Visualization* (October), 19–26. ISBN 0-89791-741-3.

CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 1998. A comparison of mesh simplification algorithms. *Computers & Graphics 22*, 1 (Feb.), 37–54. ISSN 0097-8493.

CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *IEEE Visualization 2000*.

COHEN, J., VARSHNEY, A., MANOCHA, D., TURK, G., WEBER, H., AGARWAL, P., FREDERICK P. BROOKS, J., AND WRIGHT, W. 1996. Simplification envelopes. *Proceedings of SIGGRAPH 96* (August), 119–128. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

COHEN, J. D., ALIAGA, D. G., AND ZHANG, W. 2001. Hybrid simplification: Combining multi-resolution polygon and point rendering. In *Proceedings Visualization 2001*, T. Ertl, K. Joy, and A. Varshney, Eds., IEEE Computer Society Technical Committee on Visualization and Graphics Executive Committee, 37–44.

COHEN-OR, D., LEVIN, D., AND REMEZ, O. 1999. Progressive compression of arbitrary triangular meshes. *IEEE Visualization '99* (October), 67–72. ISBN 0-7803-5897-X. Held in San Francisco, California.

CROSSNO, P., AND ANGEL, E. 1997. Isosurface extraction using particle systems. In *IEEE Visualization '97*, R. Yagel and H. Hagen, Eds., IEEE, 495–498.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. *Computer Graphics 30*, Annual Conference Series, 303–312.

DE FIGUEIREDO, L. H., DE MIRANDA GOMEZ, J., TERZOPOULOS, D., AND VELHO, L. 1992. Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface'92*, 250–257.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of SIGGRAPH 99* (August), 317–324. ISBN 0-20148-560-5. Held in Los Angeles, California.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 2000. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Graphics Interface*, 145–152.

DINH, H. Q., TURK, G., AND SLABAUGH, G. 2001. Reconstructing surfaces using anisotropic basis functions. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, IEEE Computer Society, Los Alamitos, CA, 606–613.

DONOHO, D. L. 1995. De-noising by soft-thresholding. *IEEE Transactions on Information Theory 41*, 3, 613–627.

DUCHAINEAU, M. A., WOLINSKY, M., SIGETI, D. E., MILLER, M. C., ALDRICH, C., AND MINEEV-WEINSTEIN, M. B. 1997. Roaming terrain: Real-time optimally adapting meshes. *IEEE Visualization '97* (November), 81–88. ISBN 0-58113-011-2.

DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics (TOG) 21*, 3, 257–266.

ECK, M., DEROSE, T. D., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution analysis of arbitrary meshes. *Proceedings of SIGGRAPH 95* (August), 173–182. ISBN 0-201-84776-0. Held in Los Angeles, California.

EL-SANA, J., AND VARSHNEY, A. 1999. Generalized view-dependent simplification. *Computer Graphics Forum 18*, 3 (September), 83–94. ISSN 1067-7055.

ELAD, M. 2001. On the bilateral filter and ways to improve it. *IEEE Transactions On Image Processing*.

ELDAR, Y., LINDENBAUM, M., PORAT, M., AND ZEEVI, Y. Y. 1997. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing 6*, 9, 1305–1315.

FLEISHMAN, S., COHEN-OR, D., AND LISCHINSKI, D. 2000. Automatic camera placement for image-based modeling. *Computer Graphics Forum 19*, 2 (Jun), 101–110.

FLEISHMAN, S., ALEXA, M., COHEN-OR, D., AND SILVA, C. T. 2003. Progressive point set surfaces. *ACM Transactions on Computer Graphics 22*, 4.

FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) 22*, 3, 950–953.

FLORIANI, L. D., MAGILLO, P., AND PUPPO, E. 1997. Building and traversing a surface at variable resolution. In *IEEE Visualization '97*.

FLORIANI, L. D., MAGILLO, P., AND PUPPO, E. 1998. Efficient implementation of multi-triangulations. In *Proceedings IEEE Visualization'98*, IEEE, 43–50.

GOPI, M., KRISHNAN, S., AND SILVA, C. T. 2000. Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Proceedings of the 21th European Conference Computer Graphics (Eurographics-00)*, Blackwell Publishers, Cambridge, S. Coquillart and J. Duke, David, Eds., vol. 19, 3 of *Computer Graphics Forum*, 479.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, 43–54.

GOSHTASBY, A., AND O'NEILL, W. D. 1993. Surface fitting to scattered data by a sum of gaussians. *Computer Aided Geometric Design 10*, 2 (Apr.), 143–156.

GOTSMAN, C., GUMHOLD, S., AND KOBBELT, L. 2001. Simplification and compression of 3d meshes. In *European Summer School on Principles of Multiresolution in Geometric Modelling (PRIMUS), Munich*.

GREINER, G., LOOS, J., AND WESSELINK, W. 1996. Data dependent thin plate energy and its use in interactive surface modeling. *Computer Graphics Forum 15*, 3 (August), 175–186. ISSN 1067-7055.

GROSSMAN, J. P., AND DALLY, W. J. 1998. Point sample rendering. *Eurographics Rendering Workshop 1998* (June), 181–192. ISBN 3-211-83213-0. Held in Vienna, Austria.

GUEZIEC, A., TAUBIN, G., LAZARUS, F., AND HORN, W. 1998. Converting sets of polygons to manifold surfaces by cutting and stitching. In *Proceedings of the conference on Visualization '98*, IEEE Computer Society Press, 383–390.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH 99*, 325–334.

GUSKOV, I., VIDIMCE, K., SWELDENS, W., AND SCHRÖDER, P. 2000. Normal meshes. *Proceedings of SIGGRAPH 2000* (July), 95–102. ISBN 1-58113-208-5.

HART, J. C. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 10, 527–545.

HE, T., HONG, L., VARSHNEY, A., AND WANG, S. W. 1996. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (June). ISSN 1077-2626.

HEBERT, P., LAURENDEAU, D., AND POUSSART, D. 1993. Scene reconstruction and description: Geometric primitive extraction from multiple view scattered data. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 286–293.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. *Computer Graphics 26*, 2 (July), 71–78.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. *Proceedings of SIGGRAPH 93* (August), 19–26. ISBN 0-201-58889-7. Held in Anaheim, California.

HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., MCDONALD, J., SCHWEITZER, J., AND STUETZLE, W. 1994. Piecewise smooth surface reconstruction. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, 295–302.

HOPPE, H. 1996. Progressive meshes. *Proceedings of SIGGRAPH 96* (August), 99–108. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

IGARASHI, T., AND HUGHES, J. F. 2003. Smooth meshes for sketch-based freeform modeling. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, ACM Press, 139–142.

JONES, T., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*.

KALAIAH, A., AND VARSHNEY, A. 2001. Differential point rendering. In *Rendering Techniques*, Springer-Verlag, S. J. Gortler and K. Myszkowski, Eds.

KAUFMAN, A., COHEN, D., AND YAGEL, R. 1993. Volume graphics. *IEEE Computer 26*, 7 (July), 51–64.

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive geometry compression. *Proceedings of SIGGRAPH 2000*, 271–278.

KOBBELT, L., CAMPAGNA, S., AND SEIDEL, H.-P. 1998. A general framework for mesh decimation. *Graphics Interface '98* (June), 43–50. ISBN 0-9695338-6-1.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings of SIGGRAPH 98* (July), 105–114. ISBN 0-89791-999-8. Held in Orlando, Florida.

KOBBELT, L. P., VORSATZ, J., LABSIK, U., AND SEIDEL, H.-P. 1999. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum 18*, 3 (September), 119–130. ISSN 1067-7055.

KOBBELT, L. P., BAREUTHER, T., AND SEIDEL, H.-P. 2000. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum 19*, 3 (August), 249–260. ISSN 1067-7055.

KRISHNAMURTHY, V., AND LEVOY, M. 1996. Fitting smooth surfaces to dense polygon meshes. *Computer Graphics 30*, Annual Conference Series, 313–324.

KUMAR, S., MANOCHA, D., GARRETT, W., AND LIN, M. 1996. Hierarchical back-face computation. In *Eurographics Rendering Workshop 1996*, Springer Wien, New York City, NY, X. Pueyo and P. Schröder, Eds., Eurographics, 235–244. ISBN 3-211-82883-4.

LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. Maps: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98* (July), 95–104. ISBN 0-89791-999-8. Held in Orlando, Florida.

LEE, A., MORETON, H., AND HOPPE, H. 2000. Displaced subdivision surfaces. *Proceedings of SIGGRAPH 2000* (July), 85–94. ISBN 1-58113-208-5.

LEE, I. K. 2000. Curve reconstruction from unorganized points. *Computer Aided Geometric Design 17*, 2 (February), 161–177.

LEI, Z., BLANE, M. M., AND COOPER, D. B. 1996. 3L fitting of higher degree implicit polynomials. In *Proceedings of Third IEEE Workshop on Applications of Computer Vision*.

LEVIN, D. 1998. The approximation power of moving least-squares. *Mathematics of Computation 67*, 224.

LEVIN, D. 2003. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, G. Brunnett, B. Hamann, and H. Mueller, Eds. Spinger-Verlag.

LEVOY, M., AND WHITTED, T. 1985. The use of points as a display primitive. Tech. Rep. 85-022, UNC-Chapel Hill Computer Science, January.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital michelangelo project: 3d scanning of large statues. *Proceedings of SIGGRAPH 2000* (July), 131–144. ISBN 1-58113-208-5.

LINSEN, L. 2001. Point cloud representation. Tech. Rep. 2001-3, Fakultaet fuer Informatik, Universitaet Karlsruhe.

LLOYD, S. P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (March), 129–137.

LODHA, S. K., AND FRANKE, R. 1999. Scattered data techniques for surfaces. In *Dagstuhl Conference on Scientific Visualization*, IEEE Computer Society, H. Hagen, G. Nielson, and F. Post, Eds., 182–222.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics 21*, 4 (July), 163–169.

MARR, D. 1983. *Vision:A Computational Investigation into the Human Representation and Processing of Visual Information*. .W.H.Freeman.

MCALLISTER, L. F. N. D. K., POPESCU, V., LASTRA, A., AND MCCUE, C. 1999. Real-time rendering of real world environnements. In *Rendering Techniques '99*, Springer-Verlag Wien New York, D. Lischinski and G. W. Larson, Eds., Eurographics, 145–160.

MCMILLAN, L., AND BISHOP, G. 1995. Plenoptic modeling: an image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, 39–46.

MEDEROS, B., VELHO, L., AND FIGUEIREDO, L. 2003. Moving least squares multiresolution surface approximation. In *proceedings of SIBGRAPI 2003 - XVI Brazilian Symposium on Computer Graphics and Image*.

MORSE, B. S., YOO, T. S., RHEINGANS, P., CHEN, D. T., AND SUBRAMANIAN, K. R. 2001. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI-01)*, IEEE Computer Society, Los Alamitos, CA, B. Werner, Ed., 89–98.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Transactions on Graphics (TOG) 22*, 3, 463–470.

OKABE, A., BOOTS, B., AND SUGIHARA, K. 1992. *Spatial Tesselations — Concepts and Applications of Voronoi Diagrams*. Wiley, Chichester.

OSHER, S., AND SETHIAN, J. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79*, 12–49.

PAJAROLA, R., AND ROSSIGNAC, J. 2000. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (January - March), 79–93. ISSN 1077-2626.

PAULY, M., AND GROSS, M. 2001. Spectral processing of point-sampled geometry. *Proceedings of SIGGRAPH 2001* (August), 379–386. ISBN 1-58113-292-1.

PAULY, M., GROSS, M., AND KOBBELT, L. P. 2002. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02*.

PAULY, M., KEISER, R., KOBBELT, L. P., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (TOG) 22*, 3, 641–650.

PENG, J., STRELA, V., AND ZORIN, D. 2001. A simple algorithm for surface denoising. In *IEEE Visualization 2001*, 107–112.

PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. 2000. Surfels: Surface elements as rendering primitives. *Proceedings of SIGGRAPH 2000* (July), 335–342. ISBN 1-58113-208-5.

PRATT, V. 1987. Direct least-squares fitting of algebraic surfaces. *Computer Graphics 21*, 4 (July), 145–152.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes*, second ed. Cambridge University Press, Cambridge.

RAMAMOORTHI, R., AND ARVO, J. 1999. Creating generative models from range images. *Computer Graphics 33*, Annual Conference Series, 195–204.

RUDIN, L., OSHER, S., AND FATEMI, E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D 60*, (1–4), 259–268.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. QSplat: A multiresolution point rendering system for large meshes. In *Siggraph 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, K. Akeley, Ed., Annual Conference Series, 343–352.

RUSINKIEWICZ, S. 2001. *Real-time Acquisition and Rendering of Large 3D Models*. PhD thesis, Stanford University.

RUTISHAUSER, M., STRICKER, M., AND TROBINA, M. 1994. Merging range images of arbitrarily shaped objects. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society Press, Los Alamitos, CA, USA, 573–580.

SCHAUFLER, G., AND JENSEN, H. W. 2000. Ray tracing point sampled geometry. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering* (June), 319–328. ISBN 3-211-83535-0.

SHADE, J. W., GORTLER, S. J., HE, L.-W., AND SZELISKI, R. 1998. Layered depth images. *Computer Graphics 32*, Annual Conference Series (Aug.), 231–242.

SMITH, S. M., AND BRADY, J. M. 1997. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision 23*, 1 (May), 45–78.

SOUCY, M., AND LAURENDEAU, D. 1992. Surface modeling from dynamic integration of multiple range views. In *Proc. 11th Intl. Conf. on Pattern Recognition*, 449–452.

SRAMEK, M., AND KAUFMAN, A. E. 1999. Alias-free voxelization of geometric objects. In *IEEE Transactions on Visualization and Computer Graphics*, H. Hagen, Ed., vol. 5 (3). IEEE Computer Society, 251–267.

STUERZLINGER, W. 1999. Imaging all visible surfaces. In *Proceedings of Graphics Interface 99*, I. S. MacKenzie and J. Stewart, Eds., 115–122.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2002. Anisotropic geometric diffusion in surface processing. In *IEEE Visualization 2002*.

TAUBIN, G., GUEZIEC, A., HORN, W., AND LAZARUS, F. 1998. Progressive forest split compression. *Proceedings of SIGGRAPH 98* (July), 123–132. ISBN 0-89791-999-8. Held in Orlando, Florida.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 95*, 351–358.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *ICCV*, 839–846.

TURK, G., AND LEVOY, M. 1994. Zippered polygon meshes from range images. *Computer Graphics 28*, Annual Conference Series (July), 311–318.

TURK, G., AND O'BRIEN, J. F. 2002. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG) 21*, 4, 855–873.

TURK, G. 1992. Re-tiling polygonal surfaces. In *Computer Graphics Proceedings, Annual Conference Series*, ACM Press / ACM SIGGRAPH, 55–64. ISBN 0-201-51585-7.

TURK, G. 2001. Texture synthesis on surfaces. *Proceedings of SIGGRAPH 2001* (August), 347–354. ISBN 1-58113-292-1.

WANG, S. W., AND KAUFMAN, A. E. 1995. Volume sculpting. In *1995 Symposium on Interactive 3D Graphics*, P. Hanrahan and J. Winget, Eds., ACM SIGGRAPH, 151–156. ISBN 0-89791-736-7.

WANG, J., AND OLIVEIRA, M. 2003. A hole filling strategy for reconstruction of smooth surfaces in range images. In *proceedings of SIBGRAPI 2003 - XVI Brazilian Symposium on Computer Graphics and Image*.

WARREN, J., AND WEIMER, H. 2001. *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann.

WHEELER, M. D., SATO, Y., AND IKEUCHI, K. 1998. Consensus surfaces for modeling 3D objects from multiple range images. In *Procedings of the Sixth International Conference on Computer Vision (ICCV-98)*, Narosa Publishing House, New Delhi, S. Chandran and U. Desai, Eds., 917–924.

WITKIN, A. P., AND HECKBERT, P. S. 1994. Using particles to ssample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, 269–277.

XIA, J. C., EL-SANA, J., AND VARSHNEY, A. 1997. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (April - June). ISSN 1077-2626.

ZHOU, Y., CHEN, B., AND KAUFMAN, A. E. 1997. Multiresolution tetrahedral framework for visualizing regular volume data. *IEEE Visualization '97* (November), 135–142. ISBN 0-58113-011-2.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97* (August), 259–268. ISBN 0-89791-896-7. Held in Los Angeles, California.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics 21*, 3 (July), 322–329. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

אוניברסיטת תל-אביב

הפקולטה למדעים מדויקים ע"ש ריימונד ובברלי סקלר
בית הספר למדעי המחשב

# משטחי אוסף נקודות

החיבור מוגש כחלק מהדרישות
לקבלת תואר "דוקטור לפילוסופיה"
ע"י

שחר פליישמן

העבודה נעשתה בהנחיתו של
פרופסור דניאל כהן-אור

הוגש לסנאט של אוניברסיטת תל-אביב
אוקטובר 2003

# תקציר

בעבודה זו נתאר ייצוג למשטח המורכב מאוסף נקודות במרחב. קבוצת נקודות זו יכולה להיות הפלט הגולמי של סורק תלת מימד או תיאור פני השטח של גוף הנוצר באופן אינטראקטיבי. כבר לפני עשרים שנה הוצע לייצג גופים ע"י נקודות במרחב, אולם ייצוג זה לא היה בשימוש נרחב עד לאחרונה. חידושים בתחום הסריקה התלת ממדית וכח המחשוב הגדל של מחשבים מודרניים החיו את התחום של ייצוג גופים ע"י נקודות. רוב האלגוריתמים בהם משתמשים בנקודות לייצוג גופים מתארים את פני השטח של גוף ע"י אוסף דסקיות. בכדי להציג באיכות גבוהה גופים המתוארים בצורה זו, יש צורך לבצע סינון של אוסף הנקודות במרחב האובייקט או במרחב התמונה. עבודה זו מתמקדת באספקטים השונים של ייצוג גיאומטרי של גופים המורכבים מנקודות.

אנו מתארים ייצוג חדש לגופים מאוסף נקודות, אותו אנו מכנים "משטחי אוסף נקודות" (PSS). המשטח המתקבל הוא חלק באופן גלובלי ומחושב באופן מקומי. ייצוג זה משלב את יתרונות המודלים המורכבים מנקודות עם מודל עם בסיס מתמטי בעל תכונות גיאומטריות של רציפות וחלקות.

## משטחי אוסף נקודות

"משטחי אוסף נקודות" מוגדרים ע"י אופרטור הטלה. בהינתן נקודה הנמצאת בקרבת המשטח, אופרטור ההטלה מזיז אותה אל המשטח; המשטח מוגדר ע"י אוסף הנקודות המוטלות את עצמן. אופרטור ההטלה מתבסס על שיטת הריבועים הפחותים המוזזים (MLS) שבו מותאמת פונקציה לכל נקודה ונקודה. בניגוד לייצוג פרמטרי של משטחים שבהם מותאמת פונקציה עבור אזור על פני השטח של הגוף, ורציפות מושגת ע"י בניית פונקציות בעלות הרציפות הרצויה בשפה, בשיטת ה-(MLS) הפונקציה המותאמת בנקודה היא קומבינציה ליניאריות של פונקציות חלקות. חישוב אופרטור ההטלה מתבצע בשני שלבים: בשלב ראשון מותאם מישור לקבוצת הנקודות השכנות לנקודה אותה מטילים. בשלב שני, מותאם לקבוצת הנקודות השכנות פולינום מדרגה נמוכה. התוצאה היא משטח בעל חלקות $C^\infty$ באופן גלובלי. בניגוד לייצוג גופים בעזרת פונקציות סתומות, חישוב אופרטור ההטלה הוא מקומי בלבד, ובניגוד לאינטרפולציה בעזרת פונקציות בסיס רדיאליות, החישוב המבוצע הוא מקומי.

בחישוב אופרטור ההטלה ניתנות משקלות לנקודות שכנות בעזרת פונקצית משקל חלקה. משטח ה-PSS יכול לקרב את אוסף הנקודות או לבצע אינטרפולציה שלהן. עבור

קלט המכיל רעש או אם המשתמש מעוניין להחליק את המשטח, נגדיר פונקצית משקל מקרבת. רוחב הפונקציה שולט על מידת ההחלקה שמתבצעת. עבור קלט ללא רעשים נגדיר פונקצית משקל שתבצע אינטרפולציה. את התכונות הגיאומטריות הפנימיות של ה-PSS בנקודה כגון נורמל ועקמומיות ניתן לקרב בעזרת אותן התכונות של הפולינום שהותאם לנקודה.

אנחנו מציגים כלים לדגימה מחדש של ה-PSS. בכדי להציג באיכות גבוהה גוף על המסך, ניתן לדגום מחדש את ה-PSS ברזולוצית המסך, תוך כדי חישוב המיקום והנורמל של הנקודות.

חלק זה של העבודה הוצג ב *IEEE Visualization 2001* וגרסת ז'ורנל פורסמה ב *IEEE Transactions on Computer Graphics and Visualization*.

## ייצוג פרוגרסיבי של משטחי אוסף נקודות

ייצוג פרוגרסיבי של משטחי אוסף נקודות PPSS הוא שיטת מידול גיאומטרי המייצגת גוף ברמות פירוט: מהגס לעדין ומגוף חלק לגוף בעל פרטים. אנו בונים ייצוג פרוגרסיבי של PSS ע"י צמצום מספר הנקודות בגוף הקלט ליצירת גוף בסיס שהוא חלק יותר. בשלב שני אנחנו מעדנים את גוף הבסיס ע"י הוספת נקודות חדשות ותיקון מיקומן. תהליך העידון מוסיף נקודות בקרבת גוף הבסיס ללא מידע חיצוני, ומטיל אותן על גוף הבסיס ועל גוף הקלט. וקטור התיקון יהיה שווה להפרש בין שתי הנקודות המוטלות.

את וקטור התיקון אני מפרידים לחלק משיק וחלק בכוון הנורמל. החלק המשיק מחושב ללא מידע חיצוני בעזרת כללי עידון, והחלק בכוון הנורמל הוא החלק היחיד אותו יש לקודד. בכדי ליצור קידוד יעיל של אותו מרכיב, נשתמש באופרטור הטלה מקורב המטיל נקודה על גוף הקלט בעזרת המישור הפרמטרי שחושב לגוף הבסיס, וכך נוכל לקודד את וקטור התיקון בעזרת מספר יחיד. התוצאה היא יצוג דחוס של המודל הפרוגרסיבי.

חלק זה של העבודה התפרסם ב *ACM Transactions on Graphics*.

## אלגוריתם בילטרלי להורדת רעשים ממשטחים

בפרק זה נציג אלגוריתם להורדת רעשים ממשטחים המשמר פרטים. בניגוד להורדת רעשים בעזרת ה PSS, אלגוריתם זה משמר פרטים שאינם חלקים כגון קצוות ופינות. באל-גוריתמים של סטטיסטיקה רובסטית מחשבים ערכים סטטיסטיים תוך ניסיון להתעלם מנקודות דגימה חריגות. באלגוריתם זה נוריד רעשים מגופים כך ששני קדקודים של הגוף משתי צידיה של שפה (edge) יתייחסו אחד אל השני כאל דגימות חריגות. נתבסס על אלגוריתם הפילטר הבילטרלי להורדת רעשים מתמונות, שאותו אנו מתאימים לגופים. ההתאמה מבוססת על רעיון מתוך חישוב אופרטור ההטלה: לכל קדקוד נבנה משטח פרמ-טרי מקומי ונטיל את שכניו על אותו משטח פרמטרי; השימוש במשטח הפרמטרי מאפשר לפרק את הקדקודים בסביבה למרכיב משיק ומרכיב בכוון הנורמל. בשלב הבא נפעיל את האלגוריתם הבילטרלי מעל המישור הפרמטרי, לקבלת ערך חדש בכוון הנורמל לקדקוד

אותו מחשבים.

בנוסף, אנחנו מראים כיצד לבחור פרמטרים לאלגוריתם באופן אינטראקטיבי. המשתמש מסמן קדקוד בחלק של המודל המכיל רעש ואמור להיות נקי מרעשים ובוחר רדיוס השפעה של הקדקוד. שאר הפרמטרים מחושבים באופן אוטומטי מתוך סביבת הקודקוד שנבחר.

חלק זה הוצג ב *SIGGRAPH 2003* אשר פורסם בגליון מיוחד של *ACM Transactions on Graphics*.