Eurographics Symposium on Geometry Processing (2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

Approximating and Intersecting Surfaces from Points

Anders Adamson† and Marc Alexa‡

Department of Computer Science, Darmstadt University of Technology, Germany

Abstract

Point sets become an increasingly popular shape representation. Most shape processing and rendering tasks require the approximation of a continuous surface from the point data. We present a surface approximation that is motivated by an efficient iterative ray intersection computation. On each point on a ray, a local normal direction is estimated as the direction of smallest weighted co-variances of the points. The normal direction is used to build a local polynomial approximation to the surface, which is then intersected with the ray. The distance to the polynomials essentially defines a distance field, whose zero-set is computed by repeated ray intersection. Requiring the distance field to be smooth leads to an intuitive and natural sampling criterion, namely, that normals derived from the weighted co-variances are well defined in a tubular neighborhood of the surface. For certain, well-chosen weight functions we can show that well-sampled surfaces lead to smooth distance fields with non-zero gradients and, thus, the surface is a continuously differentiable manifold. We detail spatial data structures and efficient algorithms to compute ray-surface intersections for fast ray casting and ray tracing of the surface.

Categories and Subject Descriptors (according to ACM CCS): G.1.2 [Numerical Analysis]: Approximation of surfaces and contours I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations I.3.7 [Computer Graphics]: Ray Tracing

1. Introduction

Points samples without additional topological information gain popularity as a shape representation. On one hand, many shapes are nowadays created using sampling[33, 41], where the sampling process provides only partial connectivity information. On the other hand, points are a reasonable display primitive for shapes with high geometric or textural complexity relative to the rastered image[38, 42, 28, 49]. Since acquisition and rendering are point-based, it seems logical to stay within the framework of point-based shape representation also during the modeling stage of shapes[22, 48, 37].

Modeling or processing shapes, however, requires to interrogate the surface. For point representation this typically means to attach a continuous surface approximation to the points. Approximation of surfaces (and not just functions over a Euclidean domain) from irregularly spaced points is still a fairly young topic, where many approaches are rather practical and provide no guarantee that the reconstructed surface is, for example, continuous, manifold, or resembles the topology of the sampled surface. Interestingly, only few attempts have been made to give a criterion for sufficient sampling of a surface – a notable exception is the line of work initiated by Amenta and co-workers[4, 5, 7] Here we present a scheme for the approximation of *smooth* surfaces from irregularly sampled points that also allows formulating a sampling criterion, however, not yet as concise as Amenta's.

Our approach resembles a ray tracing technique[1] for Point Set Surfaces[32, 2, 3], a surface approximation that uses a non-linear projection operation to define the surface as the stationary points of the projection. It has been conjectured that the projection operation gives rise to a continuous manifold reconstruction. In an attempt to speed up the ray intersection computation, we have replaced the non-linear projection during ray intersection with a simpler method. We found that the surface that is implicitly defined by this operation has, in fact, comparable properties – only they are easier to prove. The requirements for the reconstruction being a continuous manifold lead to a natural and intuitive sampling criterion.

† aadamson@gris.informatik.tu-darmstadt.de
‡ alexa@informatik.tu-darmstadt.de

After establishing some context by briefly discussing related work (Section 2), we first provide the theory using a slightly less general version of our surface definition (Section 3) and then explain the iterative procedure (Section 4) and spatial data structures (Section 5).

2. Related Work

We concentrate on work that is directly related to our approach, namely, the approximation of a surface from point samples and interrogating this surface by means of fast ray-surface intersections.

Most approximation techniques define the surface implicitly, either by defining a scalar function of space or by certain constructive means.

An interesting line of approximation algorithms define the surface as a subgraph of the Delaunay complex of the points[12, 17]. Many algorithms follow this spirit and differ mostly in how they identify triangles that belong to the surface. Crust[4, 5, 7] uses vertices of the Voronoi diagram as an approximation to the medial axis. The Delaunay triangulation including these vertices connects points on the surface either to the medial axis or to natural neighbors, which allows identifying surface triangles. The connection to the medial axis leads naturally to a minimal sampling density that is linear in the proximity of the surface to the medial axis. Sufficient sampling guarantees a reconstruction of the original topology. Cocone[6, 14, 15] has similar guarantees but eliminates the step of adding Voronoi vertices to the point set. This makes the Delaunay-complex significantly smaller and, thus, the reconstruction faster. Still, constructing the Delaunay complex of millions of points is costly and some algorithm rather use local triangulations of the points[11, 20]

Hoppe et al.[25] defines an implicit function that is interestingly in a sense dual to Delaunay-type reconstruction: For each point a normal direction is estimated from neighboring points and all normals are oriented consistently. The signed distance to the surface is defined as the normal component of the distance to the closest point. Thus, the surface consists of planes through the points bounded by the Voronoi cells of the points.

In many practical cases one has multiple point samples for the same region. A way to consolidate this information is to build a distance function in a volumetric grid by properly weighting the points[13, 47].

Defining the surface as a set of planar pieces results in C^0 approximations. To achieve smoother approximations one could either build a smooth surface over the triangulation[24] or fit smoother functions. A global and smooth interpolant for scattered data can be constructed using radial basis functions (RBF). For surface approximation an implicit function is computed using extra points away from the surface[43, 45]. The computation traditionally involves the solution of a large linear system, however, is nowadays tackled using compactly supported functions[46], mulitpole expansions[8, 10, 9], thinning[18, 19, 26, 16], or hierachical clustering[27, 36].

Another approch is to fit globally smooth functions locally[21], or to perform purely local fits[39, 30] and blend these local surface approximations together[35]. The moving least squares (MLS)[31] approximation takes this approach to the extreme by building a local fit for every point on the surface. Using MLS allows defining a projection operation that defines the surface implicitly as its stationary points[32]. The projection operation could be used for resampling the surface[2, 3]. Our surface definition results from simplifying a ray intersection procedure[1] for this type of surface.

For modeling and rendering a surface has to be interrogated. While for rendering one could use the the existing points[38, 42, 28, 49], modeling typically requires operations such as ray-surface intersection, for example, to specify points on the surface by clicking. For a variety of deformation and CSG operations the MLS projection operator could be used[37].

Computing ray-surface intersections for an implicit surface is conceptually simple: The ray is substituted in the implicit surface definition. Computing the intersection is, thus, equivalent to finding a root of a function in one unknown. To speed up the intersection computation one should exploit properties of the implicit function. A common way is to compute a (local) Lipschitz constant, which yields a conservative step width[29, 23]. Another approach is to use interval analysis[34].

Schaufler and Jensen define a ray-surface intersection for point sets directly, without an intermediate surface definition[44]. They collect points within a cylinder around the ray and compute a weighted average surface location. This is very fast, however, the geometry resulting from ray surface intersection depends on the particular rays used for intersecting the surface.

3. Foundations – Simple Surface Definition and Sampling Criterion

We assume that a set of points implicitly defines a smooth manifold surface. More specifically, let points $p_i \in \mathbb{R}^3, i \in \{1, \ldots, N\}$, be sampled from a surface S (possibly with a measurement noise). The general idea of our surface definition is inspired by MLS approximation – the surface is approximated by building local polynomial approximations everywhere in space and a point $\mathbf{s}$ in space belongs to the surface if its local polynomial approximation contains $\mathbf{s}$. For reasons of clarity we first describe a slightly simplified version of the definition. We feel this makes the connection to the sampling criterion and the resulting properties easier to establish. The more general surface definition is given later together with an algorithm that computes ray surface intersections.

The main tools for the definition of the surface are weighted averages and weighted co-variances of the points in a tubular neighborhood around the surface. A weight function $\theta : \mathbb{R} \to \mathbb{R}$ specifies the influence of a point $\mathbf{p}_i$ using the euclidean distance, i.e. $\theta_i(\mathbf{x}) = \theta(\|\mathbf{p_i} - \mathbf{x}\|)$. Weight functions are assumed to be smooth, positive, and monotonically deacreasing (have positive first derivative).

The weighted average of points at a location $\mathbf{s}$ in space is

$$\mathbf{a}(\mathbf{s}) = \frac{\sum_{i=0}^{N-1} \theta_i(\mathbf{s})\mathbf{p}_i}{\sum_{i=0}^{N-1} \theta_i(\mathbf{s})} \tag{1}$$

and the weighted co-variance at $\mathbf{s}$ in direction $\mathbf{n}$ describes how well a plane $\mathbf{n}(\mathbf{s}-\mathbf{x}) = 0$ fits the weighted points:

$$\sigma_n^2(\mathbf{s}) = \frac{\sum_{i=0}^{N-1} \theta_i(\mathbf{s}) \left(\mathbf{n}(\mathbf{s}-\mathbf{p}_i)\right)^2}{\sum_{i=0}^{N-1} \theta_i(\mathbf{s})} \tag{2}$$

Let $\sigma(\mathbf{s})$ be the vector of weighted co-variances along the directions of the canonical base

$$\sigma(\mathbf{s}) = \begin{pmatrix} \sigma_{(1,0,0)}(\mathbf{s}) \\ \sigma_{(0,1,0)}(\mathbf{s}) \\ \sigma_{(0,0,1)}(\mathbf{s}) \end{pmatrix} \tag{3}$$

then the major axes (i.e. directions of smallest and largest weighted co-variance at a point $\mathbf{s}$) are accesible as the eigenvectors of the bilinear form

$$\Sigma(\mathbf{s}) = \sigma(\mathbf{s})\sigma(\mathbf{s})^T \tag{4}$$

where an eigenvalue is the co-variance along the direction of the associated eigenvector.

Our computation and definition of the surface mainly depends on local frames, which are built from locally estimated normals.

Definition 1 The *normal direction* $\mathbf{n}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^3$ (or *normal* for short) is defined as a direction of smallest weighted co-variance, i.e. $\min_n \sigma_n^2(\mathbf{x})$. If n is unique the normal is *well-defined*.

It is clear that the normal in $\mathbf{x}$ is given as the eigenvector of the co-variance matrix $\Sigma(\mathbf{x})$ corresponding to the smallest eigenvector. The normal is well-defined exactly if $\Sigma(\mathbf{x})$ has an eigenvalue that is strictly smaller than all other eigenvalues.

We define the surface implicitly based on normal directions and weighted averages. The implicit function $f : \mathbb{R}^3 \to \mathbb{R}$ describes the distance of a point $\mathbf{x}$ to the weighted average $\mathbf{a}(\mathbf{x})$ projected along the normal direction $\mathbf{n}(\mathbf{x})$:

$$f(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{a}(\mathbf{x}) - \mathbf{x}) \tag{5}$$

As always, the approximated surface is defined as the zero-set of the implicit function, i.e.

$$\mathcal{S} = \{\mathbf{x} : f(\mathbf{x}) = 0\} \tag{6}$$

We know from differential geometry that $\mathcal{S}$ is a smooth

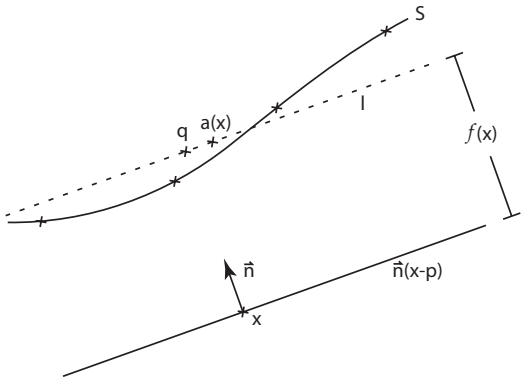

Figure 1: *The surface is defined implicitly as the zero set of a function $f(\mathbf{x})$. In each point $\mathbf{x}$ a local normal direction $\mathbf{n}(\mathbf{x})$ is estimated as the direction of minimal weighted co-variance. The implicit function $f(\mathbf{x})$ describes the distance of a weighted average $\mathbf{a}(\mathbf{x})$ of the points along normal direction.*

differentiable manifold if f is a smooth function with non-zero gradient at least in an ε-tubular neighborhood around the zero-set. Requiring f to be smooth leads to a surprisingly simple and natural sampling criterion:

Definition 2 A surface S is *well-sampled* with points $\{\mathbf{p}_i\}$ and approximated by $\mathcal{S}$ if the normals are well-defined inside a tubular neighborhood around $\mathcal{S}$.

We will show that this condition is sufficient for f being smooth for points $\mathbf{x}$ inside the tubular neighborhood: First note that f is a smooth function in $\mathbf{a}$ and $\mathbf{n}$. If all weight functions are smooth, the weighted average $\mathbf{a}$ and the weighted co-variance matrix are smooth functions in $\mathbf{x}$. Furthermore, eigenvalues are smooth functions in the matrix coefficients and eigenvectors are the solution of a linear system in the eigenvalues and the matrix. Since the normal direction is defined as the eigenvector corresponding to the smallest eigenvalue, $\mathbf{n}(\mathbf{x})$ is smooth in $\mathbf{x}$ as long as one eigenvector is always associated with the smallest eigenvalue. This has to be the case if the smallest eigenvalue is always strictly smaller than all other eigenvalues, i.e., if all normals are well-defined.

The topology of the approximated surface $\mathcal{S}$ depends on the particular choice of weight functions θ_i. It is clear that certain θ_i could lead to non-manifold approximations and that even if $\mathcal{S}$ is manifold it's not necessarily homoemorphic to S. On the other hand, weights could be so chosen that $\mathcal{S}$ is manifold and, with further restrictions, resembles the topology of S.

To give an intuition for this, we first consider an infinitesimal ball B around $\mathbf{s}$. Inside the ball, weights are constant and so are $\mathbf{a}(\mathbf{x})$ and $\mathbf{n}(\mathbf{x})$. Thus, for this region f describes a plane and if this plane passes the ball the approximation of

$\mathcal{S}$ inside B is a disk. If, furthermore, the gradient of f inside such ball is non-zero, $\mathcal{S}$ is manifold. We show in the Appendix that using Gaussians as weight functions is sufficient for non-zero gradient at $f(\mathbf{x}) = 0$.

For homeomorphic reconstruction the support of the weights should be so chosen that they separate different sheets of the surface (note that sufficient sampling is a prerequisite for differentiation of sheets). As Gaussians have inifinite support we have no practical means to construct theoretically correct weights yet, however, in practice Gaussians with approriate radius perform quite nicely (as is demonstrated in Section 6).

4. Ray-surface intersections

The surface definition given in the previous section implies a technique to efficiently compute ray-surface intersections. The idea is to evaluate function f, as it provides a rough approximation of the distance field to $\mathcal{S}$ for $f(x) \neq 0$. For fixed **n** and **a** Equation 5 describes the planar fit

$$l(\mathbf{x}) = \mathbf{n}(\mathbf{s}) \cdot (\mathbf{a}(\mathbf{s}) - \mathbf{x}) \tag{7}$$

to S with respect to the location **s** (depicted by the dashed line in Figure 1). The smaller the distance $f(\mathbf{s})$ is, the better S is approximated in **q**, which is the projection of x onto l along n. If $f(\mathbf{s}) = 0$, $\mathbf{q} = \mathbf{s}$ is a point on the surface $\mathcal{S}$.

The approximation l is used to converge to S along a ray r, using an iterative scheme similar to the ray tracing approach for MLS-surfaces[1]. Once an approximation is determined, the equation $r = r_o + s \cdot t_d$ is inserted in $l(x)$, and solving $l(\mathbf{r}) = 0$ provides t_d for the intersection of ray and planar surface approximation. The series of intersections $\{r_i\}$ approaches the surface $\mathcal{S}$. In theory, once $\mathbf{r}_i$ is close enough to the surface, the series $|f(\mathbf{r}_i)|, |f(\mathbf{r}_{i+1})|, |f(\mathbf{r}_{i+2})|, \ldots$ is strictly descreasing and an increase could be used to bail out off the iteration and start over. In practice, however, we have to accomodate imperfect weighting and use more tolerant iteration conditions: We require that the closest point **q** on the approximated surface is close to the current position on the ray $\mathbf{r}_i$ that has been used to compute **q** as well as close to the next intersection $\mathbf{r}_{i+1}$ of the ray and the plane $l = 0$ through **q**. Specifically, a region of trust T around **q** has to contain $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$. If not, the iteration is terminated and no ray surface intrsection in the proximity of $\mathbf{r}_0$ could be reported.

How to obtain an initial point $\mathbf{r}_0$ is discussed later in Section 5. The region T depends on the weighting-function for $n(x)$ and $a(x)$, which affects the size of features in $\mathcal{S}$. How to make suitable choices for the weighting-function and T is also addressed in Section 5.

The procedure sketched above is easily generalized by taking the following point of view: In each point on **r** a local coordinate system is created using the approximation of a normal direction. In this coordinate system a weighted least square *constant* approximation to the surface is computed

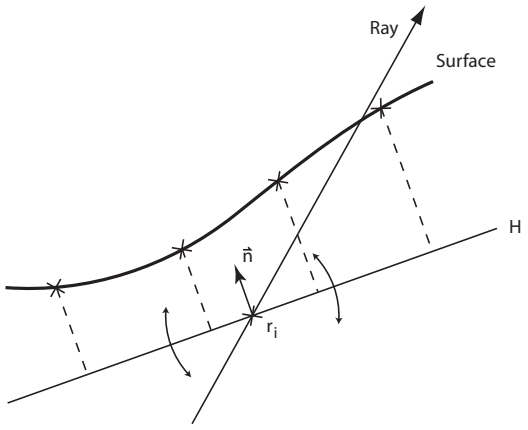

Figure 2: *In an intermediate approximation of the ray surface intersection $\mathbf{r}_i$ a local coordinate system H is established using the direction of minimal weighted co-variance as the normal* **n**

(the weighted average is a constant least squares approximation). This local approximation is intersected with the ray and the procedure is repeated. In this setting it seems quite natural to use higher order least squares approximations to the surface in the local coordinate system in an attempt to increase the approximation order of the scheme. In retrospect, we have used constant polynomials for the analysis because this allows explicitly solving the linear system that determines the polynomial.

The following three steps are iterated until the sequence $\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \ldots$ converges to the ray-surface intersection or the procedure is terminated, focussing on the next region to be examined:

1. **Support plane:** The normal of a support plane H in $\mathbf{r}_i$ is determined by minimizing the weighted distances of the points $\mathbf{p}_j$ to H. The weights are computed from the distances of the $\mathbf{p}_j$ to $\mathbf{r}_i$ using a smooth, positive, monotone decreasing function θ (e.g. a gaussian $\theta(d) = e^{-\frac{d^2}{h^2}}$). This weighted least squares problem is solved by minimizing

$$\sum_{j=1}^{N} \big(\langle \mathbf{n}, \mathbf{p}_j - \mathbf{r}_i \rangle\big)^2 \theta\big(\|\mathbf{p}_j - \mathbf{r}_i\|\big), \tag{8}$$

The minimization of equation 8 can be rewritten in bilinear form

$$\min_{\|\mathbf{n}\|=1} \mathbf{n}^T B \mathbf{n}, \tag{9}$$

where $B = \{b_{kl}\}$ is the matrix of weighted co-variances

$$b_{lk} = \sum_{j=1}^{N} \theta_j \big(p_{j_k} - r_{i_k}\big)\big(p_{j_l} - r_{i_l}\big).$$

The minimization problem in equation 9 is solved by

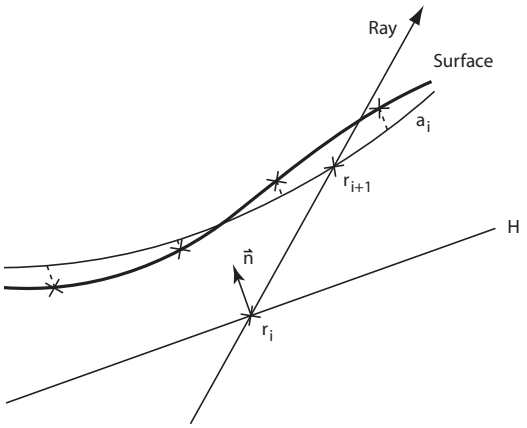

Figure 3: *The local coordinate system H is used to compute a local bivariate polynomial approximation to $\mathcal{S}$. This approximation is intersecte with* $\mathbf{r}$ *to yield the next approximation to the ray surface intersection* $\mathbf{r}_{i+1}$.

computing the eigenvector of B with the smallest eigenvalue. The resulting H is approximately parallel to the Surface in the area around its nearest approach to r_i.

2. **Polynomial approximation:** The support plane in $\mathbf{r}_i$ is used to compute a local bivariate polynomial approximation a_i of $\mathcal{S}$. To determine the coefficients of a_i, again a weighted least squares problem is solved by minimizing the equation

$$\sum_{j=1}^{N} \big(a_i(x_j,y_j) - f_j\big)^2 \theta\big(\|\mathbf{p}_j - \mathbf{r}_i\|\big), \qquad (10)$$

where (x_j,y_j) is the projection of $\mathbf{p}_j$ onto H in normal direction and $f_j = \langle \mathbf{n}, \mathbf{p}_j - \mathbf{r}_i \rangle$ is the height of $\mathbf{p}_j$ over H. Equation 10 can be minimized by calculating its gradient over the unknown coefficients of the polynomial. This leads to a system of linear equations, which is solved using standard numerical methods. The resulting polynomial is a local approximation of the surface $\mathcal{S}$. If $\mathbf{r}_i$ is sufficiently close to $\mathcal{S}$, a_i is expected be a good approximation to $\mathcal{S}$ around $\mathbf{r}_i$ and the intersection $\mathbf{r}_{i+1}$ of the ray with a_i could be used to converge to $\mathcal{S}$.

3. **Intersection:** If the ray intersects a_i, this point r_{i+1} serves as the starting point for the next iteration. If the ray misses a_i the iteration is terminated. Only ray-polynomial intersections within T are considered. An intersection is detected, when the constant part c of a_i is zero. In practice a c being smaller then a ε suffices to accept r_i (or the ray-intersection with a_i) as an adequate ray-surface intersection.

5. Spatial data structures

In this Section we describe how to represent a tubular neighborhood around the surface. This neighborhood is needed to make sure that the intersection procedure described earlier starts from a suitable point $\mathbf{r}_0$. Moreover, using simple primitives for the representation of the neighborhood significantly speeds-up ray surface intersection. In practice, the size of a tubular neighborhood around the surface that contains only well-defined normals is unknown a-priori. Our best choice is to construct a spatial region that has a certain maximum distance to the point set as we expect the distance of the points to the reconstructed surface to be bounded.

Specifically, we construct a set of balls B_i of radius ρ_i around the points p_i. If the surface $\mathcal{S}$ is contained in the union of the balls the balls are a bounding volume of $\mathcal{S}$ that is easy to test for intersection. If a ray intersects $\mathcal{S}$, it also intersects at least one ball containing the intersection. Thus, an intersected ball indicates a potential ray surface intersection. The radius h has to be chosen, to ensure that

$$\mathcal{S} \subseteq \sum_{i=0}^{N} B_i. \qquad (11)$$

Unfortunately, $\mathcal{S}$ is unknown a priori. The only a-priori knowledge are the points p_i, which are expected to be very close to the surface $\mathcal{S}$. Therefore, we choose conservative radii ρ_i, so that each B_i encloses the k-nearest neighbors of $\mathbf{p}_i$. In practice, we use $k = 6$.

The intersection of a ray with the set of balls can be computed efficiently. To quickly determine a subset of potentially intersected balls, the balls are arranged within an octree (see Figure 4), which is traversed along the ray using a parametric algorithm [40]. The current octree-voxel provides the candidate balls to be tested against the ray. Intersected balls are sorted along the ray to ensure that the first ray surface intersection is computed.

Each ray ball intersection is handled as follows: The center $\mathbf{p}_i$ of B_i is used as initial point $\mathbf{s}$ for the construction of a local coordinate system and polynomial approximation a_i. Using $\mathbf{p}_i$ for this approximation instead of the ray ball intersection has two reasons: First, $\mathbf{p}_i$ is expected to be close to $\mathcal{S}$ and should provide a reasonable approximation of the surface around $\mathbf{p}_i$. Second, the coordinate system and polynomial approximation within B_i are independent of the ray and can be stored for intersection with the next ray that intersects B_i. Intersecting the ray with a_i yields $\mathbf{r}_0$. Figure 4 illustrates this idea.

Then the procedure detailed in Section 4 is applied using B_i as the region of trust T. Thus, if the ray intersects the polynomial approximation inside B_i, the step is repeated until the desired accuracy is reached; otherwise no ray surface intersection is found within B_i and the next intersected ball along the ray is inspected.

Sometimes it is not important to determine the first intersection along a ray but only if there is any intersection with the object. A prominent example for rendering algorithms are shadow rays in ray tracing. Once a shadow-ray is

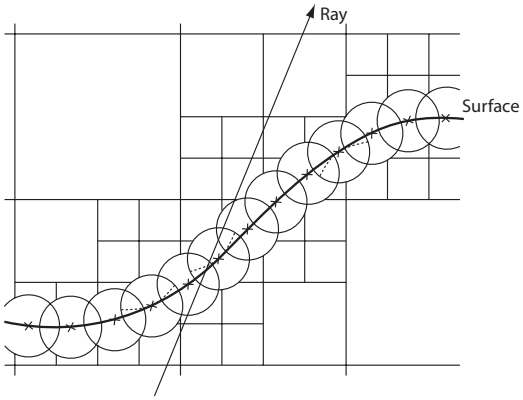

Figure 4: *The spatial data structres used to represent a tubular neighborhood around $\mathcal{S}$ is constructed as union of balls around the points. This region has the property that it contains all points with a certain maximum distance to the points and represents a best guess to the tubular neighborhood as points are expected to be close to the surface. For reducing the number of ray sphere intersections an octree is used.*

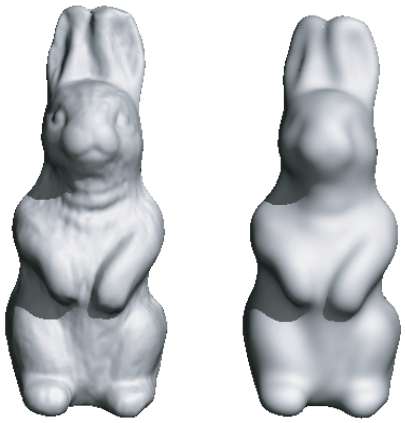

Figure 5: *The point set for analysis of the ray surface intersections, rendered using different values for the smoothing parameter h: Rabbit Sculpture - Cyberware, 67,038 points, images rendered with 200x400 pixel; left: $h = 375 \cdot 10^{-5}$, right: $h = 1700 \cdot 10^{-5}$ of the object's diameter*

obstructed by an opaque object, it is not necessary to determine at what position the ray first hits that object. Such a ray can be simply discarded from the illumination computations. This specific ray intersection query can be optimized by finding an intersection as quickly as possible anywhere on the ray. As spheres intersected close to their center are more likely to contain an intersection with the surface, they are sorted according to the distance d_i from the ray to the center c_i. The following equation determines the priority γ_i of a sphere considering different radii ρ_i.

$$\gamma_i = \frac{\rho_i - d_i}{\rho_i}. \tag{12}$$

6. Applications & Results

We have used the ray surface intersection algorithm to compute renderings by means of ray tracing. In practice, we use Gaussian weights, i.e. $\theta_i(\mathbf{x}) = \exp(\|\mathbf{x} - \mathbf{p}_i\|^2 / h^2)$. The global parameter h allows specifying the locality of the approximation. Using smaller values for h results in a more local approximation, larger values could be used to smooth out small variations in the surface (e.g. noise). Since rendering is very fast, estimating useful values for h is done interactively. Note that for uneven sampling a localized Gaussian weighting has proven to be beneficial for MLS projection operation[37]. We have found our surface to exhibit less artifacts than the surface defined from the MLS projection so that we have not yet experimented with varying values for h.

To analyze the performance of the ray intersection algorithm we have computed several renderings of the Cyberware Rabit Model† consisting of 67.037 points. Connectivity information available in the original data was discarded.

The effect of using different values for h is shown in Figure 5: the left rabbit results from using $h = 0.00375d$ and the right using $h = 0.017d$, where d is the object's diameter; as expected, larger values for h yield a smoother surface.

The following timings have been acquired using $h = 0.004d$ and an image raster of 200x400 pixel on a P4 with 2GHz: In 10.3 seconds ray surface intersections for 42,463 of the 80,000 rays were computed. In total the distance function was evaluated 238.715 times (each evaluation requires estimating a normal and computing the polynomial approximation). Roughly half of the evaluations lead to an intersection of the surface, the other half leads to bailing out of the iteration. If the center polynmials are stored and reused only 83.922 evaluations have to be calculated, where we compute and store the center polynomials on the fly.

To estimate the overhead of computing the pixel intensities and intersecting the rays with the spatial data structures we have substituted the ray surface intersection procedure with intersecting precomputed polynomials in the sphere centers. This simplification needs 1.3 seconds to ray trace the same scene. Apparently, most of the time is spent calculating and intersecting polynomial approximations.

An average 2.91 iterations were sufficient to satisfy a predefined precision of $p = 10^{-3}h$, which seems sufficient as features are expected to be larger than h. Table 1 shows the

† in reminiscence of its ubiquitous chocolate version in some countries these days

Precision (h)	10^{-1}	10^{-3}	10^{-7}	10^{-10}	10^{-11}
Avg. Iter.	1.99	2.91	4.98	6.56	10.4
Time (sec)	7.9	11.5	18.9	24.6	42.7

Table 1: *Average number of iterations until convergence to a ray surface intersection and time needed to render an image at resolution of 200x400 pixels relative to the required precision.*

average number of iterations until convergence to ray surface intersection and the time to render the whole image relative to the required precision. Increasing the precision by an order of magnitude results in about 1.5 times iterations in average. The maximum precision that could be achieved is about $10^{-10}h$. Increased the required precision further leads to a numerical breakdown of the procedure, possibly due to the eigenvector computation. This explains the superlinear number of iterations and computation time in the last column of the table.

7. Conclusions

We have presented a surface approximation technique that is based on an iterative ray-surface intersection algorithm. The definition of the surface allows deriving an intuitive criterion for sufficient sampling given a weighting function for the points. As the surface is defined by the ray intersection algorithm, ray tracing is a natural way to render the surface. Compared to ray tracing point set surfaces[1] our new approach is two orders of magnitude faster. It is comparable in speed to Schaufler & Jensen's approach[44], however, using a solid surface definition.

We admit that our formulation of the sampling criterion has several loose ends and that we are far from having a solid theory, nevertheless, we felt the results are useful and interesting. Important next steps are the definition of weights from a given smooth surface and the minimal extent of the tubular neighborhood. This would make the sampling criterion sufficient, yet still not very practical: One could only decide that a surface is not well-sampled by finding a point inside the neighborhood with undefined normal, which is very unlikely. Rather, we need conditions that necessarily lead to sufficient sampling (possibly accepting some oversampling).

Acknowledgements

The rabbit model depicted in Figure 5 is courtesy of Cyberware, the dragon model in the accompanying video is of the Stanford Computer Graphics Laboratory The early human models in Figure 6 were 3d-digitized by Peter Neugebauer of Polygon Technology Ltd, Darmstadt, Germany using a structured light scanner and the QTSculptor system.

References

1. A. Adamson and M. Alexa. Ray tracing point set surfaces. In *Proceedings of Shape Modeling International 2003*, May 2003. in press, online available at http://www.igd.fhg.de/ alexa/paper/raypss.pdf.

2. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *IEEE Visualization 2001*, pages 21–28, October 2001. ISBN 0-7803-7200-x.

3. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Computer Graphics and Visualization*, 9(1):3–15, 2003.

4. N. Amenta, M. Bern, and D. Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, March 1998.

5. N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. *Proceedings of SIGGRAPH 98*, pages 415–422, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

6. N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of the 16th Symposium on Computational Geometry*, pages 213–222, 2000.

7. N. Amenta, S. Choi, and R. Kolluri. The power crust, unions of balls, and the medial axis transform. *CGTA: Computational Geometry: Theory and Applications*, 19(2–3):127–153, 2001.

8. R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines. *IMA J. Numer. Anal.*, 17(3):343–372, 1997.

9. R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: domain decomposition methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740 (electronic), 2000.

10. R. K. Beatson and G. N. Newsam. Fast evaluation of radial basis functions: moment-based methods. *SIAM J. Sci. Comput.*, 19(5):1428–1449 (electronic), 1998.

11. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, October - December 1999. ISSN 1077-2626.

12. J.-D. Boissonnat. Geometric structues for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, October 1984.

13. B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Proceedings of SIGGRAPH 96*, pages 303–312, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

14. T. K. Dey, J. Giesen, and J. Hudson. Delaunay based shape reconstruction from large data. In *IEEE Symposium on Parallel and Large Data Visualization*, pages 19–27, 2001.

15. T. K. Dey and S. Goswami. Tight cocone: A water-tight surface reconstructor. In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications*, 2003.

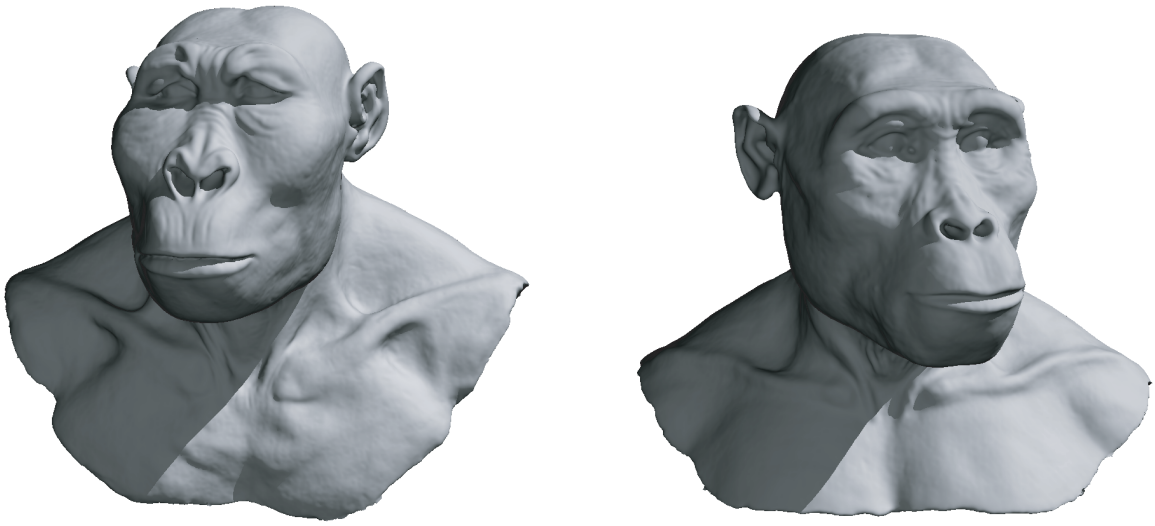

Figure 6: *Ray tracings of raw point data acquired with a structured light scanner. Both models have roughly 200K points and rendering an image with 906x868 pixel requires about 3 minutes on a P4 2GHz.*

16. N. Dyn, M. Floater, and A. Iske. Adaptive thinning for bivariate scattered data. *J. Comput. Appl. Math.*, 2002. to appear.

17. H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, January 1994. ISSN 0730-0301.

18. M. S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *J. Comput. Appl. Math.*, 73(1-2):65–78, 1996.

19. M. S. Floater and A. Iske. Thinning algorithms for scattered data interpolation. *BIT*, 38(4):705–720, 1998.

20. M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Graphics Forum*, 19(3), August 2000.

21. A. Goshtasby and W. D. O'Neill. Surface fitting to scattered data by a sum of gaussians. *Computer Aided Geometric Design*, 10(2):143–156, April 1993.

22. M. Gross. Are points the better graphics primitives? *Computer Graphics Forum*, 20(3), 2001. ISSN 1067-7055.

23. J. C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(9):527–545, 1996. ISSN 0178-2789.

24. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Proceedings of SIGGRAPH 94*, pages 295–302, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

25. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):71–78, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

26. A. Iske. Hierarchical scattered data filtering for multilevel interpolation schemes. In *Mathematical methods for curves and surfaces (Oslo, 2000)*, pages 211–221. Vanderbilt Univ. Press, Nashville, TN, 2001.

27. A. Iske and J. Levesley. Multilevel scattered data approximation by adaptive domain decomposition. Technical report, 2002. Preprint.

28. A. Kalaiah and A. Varshney. Differential point rendering. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, pages 139–150. Eurographics, June 2001. ISBN 3-211-83709-4.

29. D. Kalra and A. H. Barr. Guaranteed ray intersections with implicit surfaces. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):297–306, July 1989. Held in Boston, Massachusetts.

30. Z. Lei, M. M. Blane, and D. B. Cooper. 3L fitting of higher degree implicit polynomials. In *Proceedings of Third IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, USA, Dec. 1996.

31. D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224), 1998.

32. D. Levin. Mesh-independent surface interpolation, 2003.

33. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. *Proceedings of SIGGRAPH 2000*, pages 131–144, July 2000. ISBN 1-58113-208-5.

34. D. P. Mitchell. Robust ray intersection with interval arithmetic. *Graphics Interface '90*, pages 68–74, May 1990.

35. Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on*

Computer Graphics (SIGGRAPH 2003 Proceedings), 22(3), 2003.

36. Y. Ohtake, A. Belyaev, and H.-P. Seidel. A multi-scale approach to 3d scattered data interplation with compactly supported basis functions. In *Proceedings of Shape Modeling International 2003*, May 2003. in press.

37. M. Pauly, R. Kaiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (SIGGRAPH 2003 issue)*, 22(3), 2003. to appear.

38. H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. *Proceedings of SIGGRAPH 2000*, pages 335–342, July 2000. ISBN 1-58113-208-5.

39. V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):145–152, July 1987. Held in Anaheim, California.

40. J. Revelles, C. Urena, and M. Lastra. An efficient parametric algorithm for octree traversal. In *WSCG 2000 Conference Proceedings*, 2000.

41. S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

42. S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. *Proceedings of SIGGRAPH 2000*, pages 343–352, July 2000. ISBN 1-58113-208-5.

43. V. V. Savchenko, A. A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, October 1994.

44. G. Schaufler and H. W. Jensen. Ray tracing point sampled geometry. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 319–328, June 2000. ISBN 3-211-83535-0.

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

46. H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(4):389–396, 1995.

47. M. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *IEEE International Conference on Computer Vision 1998*, pages 917–924, 1998.

48. M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics*, 21(3):322–329, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

49. M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, July - September 2002. ISSN 1077-2626.

Appendix A: A sufficient condition for manifold reconstruction

To guarantee that $f(\mathbf{x}) = 0$ is manifold (assuming that the surface is well-sampled and, thus, f is smooth) we have to show that $\nabla f(\mathbf{x}) \neq 0$ for $f(\mathbf{x}) = 0$. We differentiate along normal direction in $\mathbf{x}$ to reduce the problem to one dimension. Note that this is possible because the surface is well-sampled, which implies that normals are well-defined. We can re-write f along this dimension as

$$f(\xi) = n(\xi)(a(\xi) - \xi) = \frac{\sum \theta_i(\xi)(\xi - \phi_i)^2}{\sum \theta_i(\xi)} \left(\frac{\sum \theta_i(\xi)\phi_i}{\sum \theta_i(\xi)} - \xi \right) \tag{13}$$

where ϕ_i denotes the distance of $\mathbf{p}_i$ along normal direction. Since we have defined differentiation w.r.t. ξ in direction of the normal n, $f(\xi) = 0$ implies $a(\xi) = 0$ so that we have to show only $a'(\xi)$ to be non-zero:

$$a'(\xi) = \frac{\sum \theta_i'(\xi)\phi_i \sum \theta_i(\xi) - \sum \theta_i(\xi)\phi_i \sum \theta_i'(\xi)}{\left(\sum \theta_i(\xi)\right)^2} - 1 \neq 0 \tag{14}$$

A particular simple way to satisfy this inequality is to use strictly positive weight functions of the form $\theta' = c\theta$ (i.e. exponential functions) because then the nominator is equal zero and the denominator is strictly positive. However, also Gaussian weight functions

$$\theta_i(h) = e^{|\xi - \phi_i|/h^2} \tag{15}$$

satisfy Eq. 14 for almost all h: The main observation is that

$$\sum \theta_i'(h)\phi_i \sum \theta_i(h) - \sum \theta_i(h)\phi_i \sum \theta_i'(h) - \sum \theta_i(h) \sum \theta_i(h) \tag{16}$$

is a smooth function in h, whose gradient cannot vanish everywhere because the Gaussians have strictly positive derivatives for positive values of h. Thus, Eq. 16 is smoothly varying with h and has only finitely many zeroes. All h that equate to non-zero values are sufficient for a local manifold reconstruction. Note that this is not sufficient for the existence of a global h that guarantees manifoldness everywhere on the surface, on the other hand, it is very unlikely in practice that the set of forbidden h-values all over the zero-set of f is dense in $\mathbb{R}$. A better characterization of admissible values for h is, nevertheless, desirable.