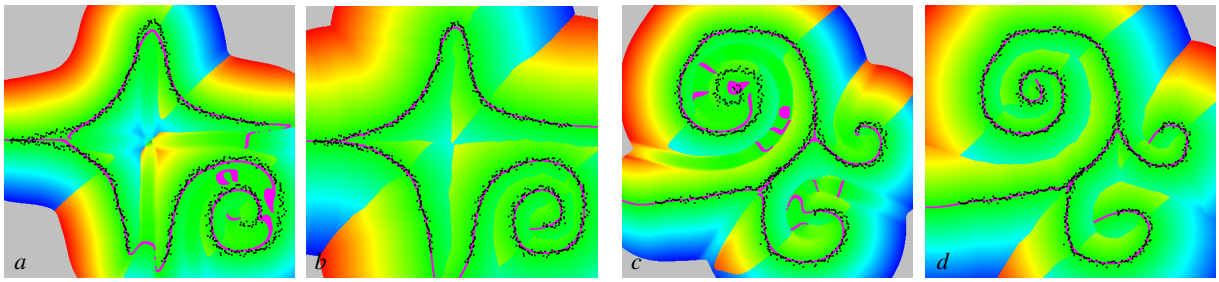
Eurographics Symposium on Point-Based Graphics (2004)
M. Alexa, S. Rusinkiewicz, (Editors)

Proximity Graphs for Defining Surfaces over Point Clouds

Jan Klein[1] and Gabriel Zachmann[2]

[1] Heinz Nixdorf Institute and Institute of Computer Science, University of Paderborn, Germany
[2] Department of Computer Science II, University of Bonn, Germany



Visualization of the moving least squares surface (magenta) over a 2D point cloud (black dots) based on different distance functions: (a,c) Euclidean, (b,d) ours based on proximity graphs.

Abstract

We present a new definition of an implicit surface over a noisy point cloud. It can be evaluated very fast, but, unlike other definitions based on the moving least squares approach, it does not suffer from artifacts.
In order to achieve robustness, we propose to use a different kernel function that approximates geodesic distances on the surface by utilizing a geometric proximity graph. The starting point in the graph is determined by approximate nearest neighbor search. From a variety of possibilities, we have examined the Delaunay graph and the sphere-of-influence graph (SIG). For both, we propose to use modifications, the r-SIG and the pruned Delaunay graph.
We have implemented our new surface definition as well as a test environment which allows to visualize and to evaluate the quality of the surfaces. We have evaluated the different surfaces induced by different proximity graphs. The results show that artifacts and the root mean square error are significantly reduced.

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

1. Introduction

In the past few years, point clouds have had a renaissance caused by the wide-spread availability of scanning technology. In order to render [PvBZG00, RL00, ZPvBG02, BWG03] and interact [KZ04] with objects thus represented, one must define an appropriate surface (even if it is not explicitly reconstructed).

This definition should produce a surface that is as close to the original surface as possible. At the same time, it should allow the object to be rendered and interacted with as fast as possible.

In this paper, we present a new definition of a surface over a given point cloud. It is based on moving least squares (MLS), i.e., it is the zero set of a function $f(\mathbf{x})$ that is based on weighted averages and MLS regression.

The simple MLS definition of point cloud surfaces is quite attractive and can be evaluated very fast. However, it suffers from artifacts in the surface. They are caused by a distance function that is not adapted to the topology of the surface: the Euclidean distance makes points "close" to $\mathbf{x}$ that are really topologically far away.

The idea of our method is to utilize (conceptually) a Voronoi diagram to find the nearest neighbor of a query point $\mathbf{x}$, and then traverse the Voronoi diagram breadth-first to compute approximate geodesic distances between the query point and the cloud points. Since the Voronoi diagram basically provides an adjacency relation based on some notion of proximity, we can also use other *proximity graphs*. Here, we investigate also the *sphere-of-influence* graph, and a generalization, that provides a natural notion of proximity in our context.

This way, our new method offers the advantages of MLS, but does not suffer from robustness issues and offers the potential to handle non-uniform point clouds.

In order to evaluate the quality of our surfaces we generate noisy point clouds from a given "exact" surface. For these, we compute the deviation of the zero sets of the different definitions from the original exact surface. The results show that our new definition produces much better surfaces. In addition, our experiments show that our method can be evaluated very fast.

2. Related Work

The representation of objects by point clouds is based on some notion of *surface* that describes the surface in-between the points, which are samples taken from an original surface, usually with error.

One way is to extend the points to so-called surfels yielding a piece-wise constant surface [RL00, PvBZG00]. Our work does not deal with this kind of surface representation.

Another way is to consider the problem of *reconstruction*, where a continuous surface is explicitly constructed from the set of points, usually in the form of a polygonal mesh. Several methods can be distinguished; an attractive one are *combinatorial methods* because they can guarantee the reconstructed mesh to be homeomorphic to the original surface under some reasonable assumptions [ASCL02,DG03]. Other methods are more cluster- or graph-based [HUHJ01, HDD*92]. With the present paper, we are not concerned with this approach, because it does not stay within the framework of point clouds.

An attractive way of handling point clouds is to define the surface as the zero set of an *implicit function* that is constructed from the point cloud. Usually, this function is not analytically but "algorithmically" given. This is a general method that can be used for reconstruction as well as ray-tracing or collision detection.

An interesting method pursuing this approach is the use of natural coordinates (which are based on Voronoi diagrams) [BC00]. They are used to turn Hoppe's discontinuous definition [HDD*92] into a continuous one. However, computing the natural coordinates is very expensive.

A very popular class of methods is to define the surface locally as the graph of a function [ABCO*03, AA03, AA04, Lev03, AK04]. For each evaluation of the function, an approximating polynomial function needs to be computed over a suitable projection plane, both of which are found using MLS. They are fairly simple to implement but difficult to make robust. In particular, non-uniform point clouds are difficult to handle generally, and there can be extra zero sets.

Recent publications have, therefore, proposed to partition the point set by an octree and fit quadratic functions only to leaves that are occupied by points [OBA*03].

As mentioned above, our method is based on proximity graphs, which have been studied extensively in the past decade. There is a broad spectrum of them, including the Delaunay graph, nearest-neighbor graph, γ-graph, α-shape, and the spheres-of-influence graph, to name but a few; see [JT92] for a good survey. They have been used for OCR [BLS00, MQ03], reconstruction [Vel93], and many other applications.

In [Lee00], a Euclidean minimum spanning tree is used to thin out a given set of unorganized points from which one can reconstruct the surface afterwards.

3. Moving Least Squares

In this section, we will first give a quick recap, and then explain the problem of the conventional MLS method. For sake of clarity, all illustrations are in 2D, but the methods work in $\mathbb{R}^3$ as well (and, in fact, in any dimension).

3.1. Surface Definition

Let a point cloud $\mathcal{P}$ with N points $\mathbf{p}_i \in \mathbb{R}^3$ be given. Then, an appealing definition of the surface from $\mathcal{P}$ is the zero set $S = \{\mathbf{x} | f(\mathbf{x}) = 0\}$ of an implicit function [Lev03, AA03]

$$f(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{a}(\mathbf{x}) - \mathbf{x}) \tag{1}$$

where $\mathbf{a}(\mathbf{x})$ is the weighted average of all points $\mathcal{P}$

$$\mathbf{a}(\mathbf{x}) = \frac{\sum_{i=1}^{N} \theta(\|\mathbf{x} - \mathbf{p}_i\|)\mathbf{p}_i}{\sum_{i=1}^{N} \theta(\|\mathbf{x} - \mathbf{p}_i\|)}. \tag{2}$$

Usually, a Gaussian kernel (weight function)

$$\theta(d) = e^{-d^2/h^2} \tag{3}$$

with $d = \|\mathbf{x} - \mathbf{p}\|$, is used, but other kernels work as well. The global bandwidth of the kernel, given by h, allows us to tune the decay of the influence of the points. It should be chosen such that no holes appear [KZ04].

The normal $\mathbf{n}(\mathbf{x})$ is determined by moving least squares. It is defined as the direction of smallest weighted covariance, i.e., it minimizes

$$\sum_{i=1}^{N} \big(\mathbf{n}(\mathbf{x}) \cdot (\mathbf{a}(\mathbf{x}) - \mathbf{p}_i)\big)^2 \theta(\|\mathbf{x} - \mathbf{p}_i\|) \tag{4}$$

for fixed $\mathbf{x}$ and under the constraint $\|\mathbf{n}(\mathbf{x})\| = 1$.

Note that, unlike [AA03], we use $\mathbf{a}(\mathbf{x})$ as the center of the PCA, which makes the function f much more aesthetically appealing (see Figure 1). Also, we do not solve a minimization problem like [Lev03, ABCO*03], because we are aiming at an extremely fast method.

The normal $\mathbf{n}(\mathbf{x})$ defined by (4) is the smallest eigenvector of the centered covariance matrix $\mathbf{B} = (b_{ij})$ with

$$b_{ij} = \sum_{k=1}^{N} \theta(\|\mathbf{x} - \mathbf{p}_k\|)(p_{k_i} - a(\mathbf{x})_i)(p_{k_j} - a(\mathbf{x})_j). \tag{5}$$

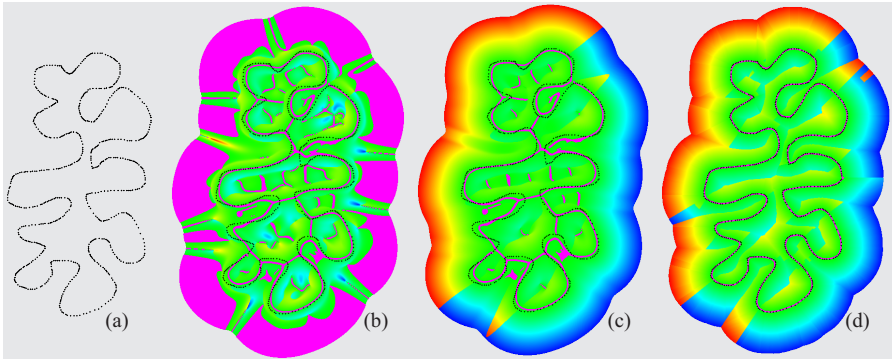


Figure 1: *Visualization of the implicit function $f(\mathbf{x})$ over a 2D point cloud. Points $\mathbf{x} \in \mathbb{R}^2$ with $f(\mathbf{x}) \approx 0$, i.e., points on or close to the surface, are shown magenta. Red denotes $f(\mathbf{x}) \gg 0$ and blue denotes $f(\mathbf{x}) \ll 0$. (a) point cloud; (b) reconstructed surface using the definition of [AA03]; (c) utilizing the centered covariance matrix produces a better surface, but it still has several artifacts; (d) surface and function $f(\mathbf{x})$ based on our more geodesic kernel using the sphere-of-influence graph.*

3.2. Euclidean Kernel

The above definition can produce artifacts in the surface S (see Figure 1); two typical cases are as follows. First, assume $\mathbf{x}$ is halfway between two (possibly unconnected) components of the point cloud; then it is still influenced by *both* parts of the point cloud, which have similar weights in Equ. 2 and 4. This can lead to an *artificial* zero subset $\subset S$ where there are no points from $\mathcal{P}$ at all. Second, let us assume that $\mathbf{x}$ is inside a cavity of the point cloud. Then, $\mathbf{a}(\mathbf{x})$ gets "drawn" closer to $\mathbf{x}$ than if the point cloud was flat. This makes the zero set *biased* towards the "outside" of the cavity, away from the true surface. In the extreme, this can lead to cancellation near the center of a spherical point cloud, where all points on the sphere have a similar weight.

This thwarts algorithms based solely on the point cloud representation, such as collision detection [KZ04] or ray-tracing [AA04].

In all of these cases, the problem is caused by the following deficiency in the kernel (3). The Euclidean distance $\|\mathbf{x}-\mathbf{p}\|$, $\mathbf{p} \in \mathcal{P}$, can be small, while the distance from $\mathbf{x}$ to the closest point on S and then along the shortest path to $\mathbf{p}$ on S (the geodesic) is quite large.

The problems mentioned above could be alleviated somewhat by restricting the surface to $\{\mathbf{x} : \|\mathbf{x}-\mathbf{a}(\mathbf{x})\| < c\}$ (since $\mathbf{a}(\mathbf{x})$ must stay within the convex hull of $\mathcal{P}$). However, this does not help in many cases involving cavities.

4. Geodesic Distance Approximation

As mentioned above, the main problems are caused by a distance function that does not take the topology of S into account. We propose to use a different distance function that is based on geodesic distances on the surface S. Unfortunately, we do not have an explicit reconstruction of S, and in many applications, we do not even want to construct one.

Therefore, we propose to utilize a geometric proximity graph where the nodes are points $\in \mathcal{P}$. In such proximity graphs, nodes $\mathbf{p}$ and $\mathbf{q}$ are connected by an edge if some geometric proximity predicate holds.

There is a variety of different proximity graphs over a set $\mathcal{P}$, for instance the Delaunay graph $\mathrm{DG}(\mathcal{P})$, the Gabriel graph, the relative nearest neighbor graph, and the nearest neighbor graph (NNG) [JT92]. The Delaunay graph is the densest one of these, while each of the others is a subgraph of the previous one. We chose to investigate one graph from the dense end of the spectrum, namely $\mathrm{DG}(\mathcal{P})$, and one from the sparse end. Clearly, the NNG is too sparse, so we chose to investigate the sphere-of-influence graph $\mathrm{SIG}(\mathcal{P})$, for which we also propose and utilize a nice generalization.

In the following, the length of an edge is the Euclidean distance $\|\mathbf{p}-\mathbf{q}\|$ (or any other metric).

4.1. Geodesic Kernel

Given a proximity graph, we compute a restricted *all pairs shortest paths* (APSP) matrix. Computing and storing the full matrix would be, of course, prohibitively expensive. Since our kernel (3) decays fairly quickly (for reasonable choices of h), we need to store only paths up to some length; the contribution of nodes in Equations 2 and 5 that are farther away can be neglected. In Section 4.2, we show that the resulting matrix can be computed and stored in $O(N)$ time and space using a simple lookup table. Therefore, we denote it just as CPSP (close-pairs shortest-paths) map.

We now define a new distance function $\|\mathbf{x}-\mathbf{p}\|_{\mathsf{geo}}$ as follows. Given some location $\mathbf{x}$, we compute its approximate nearest neighbor $\mathbf{p}^* \in \mathcal{P}$; using a simple k-d tree, this can be done in $O(\log^3 N)$ in 3D [AMN*98]. An exact nearest neighbor could be found in time $O(\log N)$ using a Delaunay hierarchy [Dev02], but this may not always be practical.

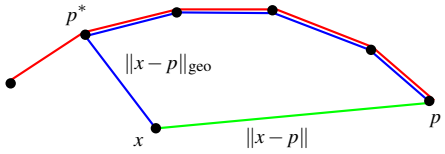


Figure 2: *Instead of the Euclidean distance $\|\mathbf{x}-\mathbf{p}\|$, we use an approximate geodesic distance $\|\mathbf{x}-\mathbf{p}\|_{\mathsf{geo}}$ based on the close-pairs shortest-paths matrix over a proximity graph.*

Starting from $\mathbf{p}^*$, we determine the distance $d(\mathbf{p}^*,\mathbf{p})$ for any $\mathbf{p} \in \mathcal{P}$ as the accumulated length of the shortest path from $\mathbf{p}^*$ to $\mathbf{p}$, multiplied by the number of "hops" along the path. This can be retrieved readily from the precomputed CPSP map. Overall, $\|\mathbf{x}-\mathbf{p}\|_{\mathsf{geo}}$ is defined by

$$\|\mathbf{x}-\mathbf{p}\|_{\mathsf{geo}} = \|\mathbf{x}-\mathbf{p}^*\| + d(\mathbf{p}^*,\mathbf{p}) \tag{6}$$

Figure 2 illustrates this idea.

The rationale for multiplying the path length by the number of hops is the following: if an (indirect) neighbor $\mathbf{p}$ is reached by a shortest path with many hops, then there are many points in $\mathcal{P}$ that should be weighted much more than $\mathbf{p}$, even if the Euclidean distance $\|\mathbf{p}^*-\mathbf{p}\|$ is small. This is independent of the concrete proximity graph used for computing the shortest paths.

Overall, when computing f by (1)–(5), we use $\|\cdots\|_{\mathsf{geo}}$ in (3). We call this modified kernel a *geodesic kernel*.

4.2. Close-Pairs Shortest-Paths Map

In this section, we show that our CPSP map can be computed and stored in $O(N)$ with $N = |\mathcal{P}|$.

Definition 1 (Sampling radius) Consider a set of spheres, centered at points $p_i \in \mathcal{P}$, that cover the surface defined by $\mathcal{P}$, where all spheres have equal radius. We define the sampling radius $r(\mathcal{P})$ as the minimal radius of such a sphere covering.

In [KZ04] it is shown, that the bandwidth h should be chosen such that points up to a distance of about $m \cdot r(\mathcal{P})$ around a point $p_i \in P$ have an influence in Equ. 1 ($m \approx 5$). That means, for each point $p_i \in \mathcal{P}$ we have to run a SSSP algorithm for all points lying in the sphere S_i with radius $m \cdot r(\mathcal{P})$ centered at p_i. The following lemma shows, that only a constant number of points is inside S_i, if $\mathcal{P}$ is a uniform (possibly noisy) sampling of a surface. As a consequence, we have to start N times a SSSP algorithm for a constant number of points. Overall, our CPSP map can be computed in time $O(N)$.

Lemma 1 Let a point cloud $\mathcal{P}$ with uniformly distributed points $\mathbf{p}_i \in \mathbb{R}^d$ ($d \in \{2,3\}$) and sampling radius $r(\mathcal{P})$ be given. Then, at most $\lceil\sqrt{d}\cdot m\rceil^d$ points $\in \mathcal{P}$ lie in a sphere with radius $m \cdot r(\mathcal{P})$.

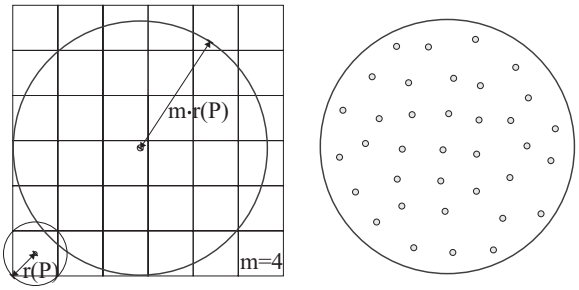


Figure 3: *Under reasonable assumptions, the close-pairs shortest-paths matrix has size $O(N)$. Left: a sphere with radius $m \cdot r(\mathcal{P})$ can be covered by $O(m^3)$ spheres with radius $r(\mathcal{P})$. Right: $\lceil\sqrt{2}\cdot m\rceil^2$ uniformly distributed points inside.*

Proof: In the following, we consider only the 3D case ($d = 3$), the 2D case can be shown analogously.

A sphere S_1 with radius $m \cdot r(\mathcal{P})$ can be covered with at most $c := \lceil\sqrt{3}\cdot m\rceil^3$ smaller spheres of radius $r(\mathcal{P})$. This has already been shown by Rogers [Rog63]: the sphere S_1 can be covered by a cube with side length $2mr(\mathcal{P})$ and the smaller spheres with radius $r(\mathcal{P})$ cover cubes with side length $\sqrt{4/3}r(\mathcal{P})$ (see Fig. 3 left). As a consequence, the larger cube can be covered by $\lceil 2mr(\mathcal{P})/\sqrt{4/3}r(\mathcal{P})\rceil^3 = c$ smaller cubes and therefore by the same number of spheres with radius $r(\mathcal{P})$.

That means, c uniformly distributed spheres of radius $r(\mathcal{P})$ with centers in S_1 cover S_1. Only if the spheres are not uniformly distributed, more than c spheres with sampling radius $r(\mathcal{P})$ are necessary to cover S_1. ■

Note that in practice often much fewer points than c lie inside S_1, in most cases $k \cdot \lceil\sqrt{d-1}\cdot m\rceil^{d-1}$ are realistic (k is a small constant of about 2 or 3).

For memory efficiency reasons, we store the CPSP matrix in a hash table of size $O(N)$ instead of using a N^2 matrix.

4.3. Proximity by Delaunay Graph

It is very intuitive to use the Delaunay graph $\mathrm{DG}(\mathcal{P})$ for our problem (which can be computed efficiently in $O(N)$ time in 3D for uniform point clouds), because [ASCL02] described an intriguing algorithm for reconstructing a polygonal surface over a point cloud without noise from the Voronoi diagram (which is the dual of the Delaunay graph). In addition, [DG04] used it as reasonable noise model. So, it is obvious that geodesic distances between the points can be approximated by shortest paths on the edges of the graph. This approximation can be improved if we also allow paths across the polygonal tessellation, not only on the edges [KS00, CH90].

Since the $\mathrm{DG}(\mathcal{P})$ induces a neighborhood relation that also includes "long distance" neighborhoods, some shortest

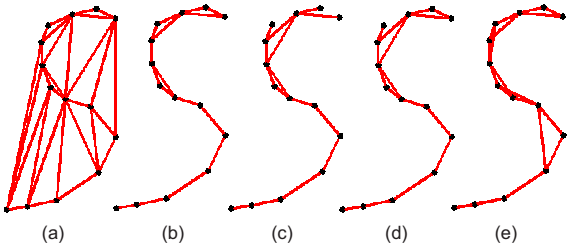


Figure 4: *Different proximity graphs. (a)* $\mathrm{DG}(\mathcal{P})$*, (b)* $\mathrm{DG}(\mathcal{P})$ *where edges are pruned according to a global sample density, (c)* $\mathrm{DG}(\mathcal{P})$*, pruning by first quartile, (d)* $\mathrm{DG}(\mathcal{P})$*, pruning by second quartile, (e)* $\mathrm{SIG}(\mathcal{P})$*.*

paths can "tunnel" through space that should really be a gap in the model (see Figure 4, left). Therefore, we prune edges from $\mathrm{DG}(\mathcal{P})$ based on criteria that involve an estimation of the local spatial density of the point cloud.

If our point cloud is well-sampled in the sense of [ASCL02], then we could prune all edges incident to a point $\mathbf{p} \in \mathcal{P}$ that are longer than the distance of $\mathbf{p}$ from the medial axis of S — *provided* we knew that distance for each $\mathbf{p}$. This is, of course, not feasible.

Therefore, we propose to utilize a statistical outlier detection method to prune edges. This is motivated by the observation that most of the unwanted "long distance" edges are local outliers, or form a cluster of outliers. In the following, we describe a simple outlier detection algorithm that seems to perform well in our case, but, of course, other outlier detection algorithms [VB94] should work as well.

In statistics, an outlier is a single observation which is far away from the rest of the data. One definition of "far away" in this context is "greater than $Q_3 + 1.5 \cdot IQR$" where Q_3 is the third quartile, and IQR is the interquartile range (equal to $Q_3 - Q_1$). As a consequence, for each node in the graph we can determine the lengths of its adjacent edges and cut edges with length of at least $Q_3 + 1.5 \cdot IQR$. However, in most cases each node in the Delaunay graph has only a handful of adjacent edges so that only few edges would be pruned. Our empirical evaluation showed us, that the best results are achieved by pruning edges with length of at least Q_2 (i.e., median).

4.4. Proximity by Sphere-of-Influence Graph

The sphere-of-influence graph (SIG) is a fairly little known proximity graph [MQ03, BLS00] which can be computed efficiently in time $O(N)$ on average for uniform point sampled models with size N in any fixed dimension [Dwy95]. The idea is to connect points if their "spheres of influence" intersect. More precisely, for each point $\mathbf{p}_i$ the distance d_i to its nearest neighbor is determined and two points $\mathbf{p}_i$ and $\mathbf{p}_j$ are connected by an edge if $\|\mathbf{p}_i - \mathbf{p}_j\| \leq d_i + d_j$.

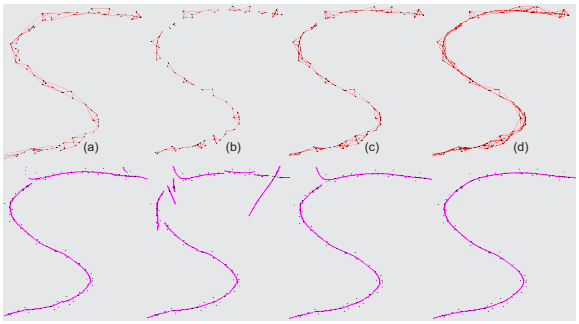


Figure 5: *If the proximity graph is too sparse, artifacts can occur. (a)* $\mathrm{DG}(\mathcal{P})$ *where edges are pruned by second quartile, (b) SIG (=1-*$\mathrm{SIG}(\mathcal{P})$*), (c) 2-*$\mathrm{SIG}(\mathcal{P})$ *(d) 3-*$\mathrm{SIG}(\mathcal{P})$*. The surface is rendered magenta.*

As a consequence, the SIG tends to connect points that are "close" to each other relative to the local point density. In contrast to the $\mathrm{DG}(\mathcal{P})$, no "long distance" neighbor relations are included, except for some pathological cases when the surface is very irregularly sampled. In these cases, we could apply once again our outlier detection algorithm proposed in the previous section.

4.5. *r*-SIG

In noisy or irregularly sampled point clouds, there can be several pairs of points that are placed much farther apart from each other than their inter-pair separation. In such situations, the $\mathrm{SIG}(\mathcal{P})$ would consist of a lot of isolated "mini-clusters", even though there are no holes in the original surface (see Figure 5 b). Consequently, the corresponding surface could not be reconstructed correctly, because the approximated geodesic distances are too imprecise: on the one hand, they are too large because points close together can only indirectly be accessed through the graph by visiting other nodes; on the other hand — in the case of unconnected components — for some points in space, too few cloud points are considered for the reconstruction.

To overcome this problem, we propose the *r-th order SIG*: instead of computing the distance to the nearest neighbor for each node, we compute the distance to the r-nearest neighbor and then proceed as in the case of $r = 1$. It is obvious that the larger r, the more nodes are directly connected by an edge, and that too large r can result in "long distance" edges as in the case of the $\mathrm{DG}(\mathcal{P})$.

4.6. Reducing Discontinuities

Independent of the proximity graph being used, there can be discontinuities in function f and sometimes even in the reconstructed surface (see Figure 6). These can occur at the borders of the Voronoi regions of the cloud points. They are

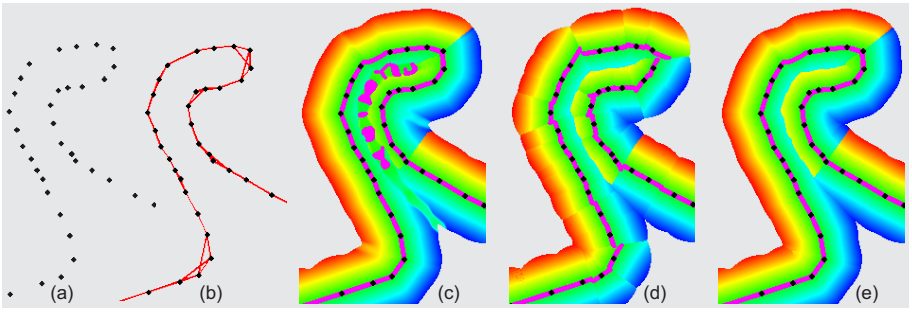


Figure 6: *Discontinuities can be avoided by our geodesic kernel* $\|\mathbf{x}-\mathbf{p}\|_{\text{geo(k)}}$. *(a) point cloud, (b) SIG, (c) Euclidean kernel, (d) geodesic kernel as of Equation 6 can cause discontinuities, (e) geodesic kernel* $\|\mathbf{x}-\mathbf{p}\|_{\text{geo(4)}}$ *as of Equation 7 causes no noticeable artifacts or discontinuities in the surface.*

more pronounced at borders where the Voronoi sites are far apart from each other, such as those close to the medial axis.

To overcome this problem, we propose to modify our geodesic kernel of Equation 6 to use a small set of k-nearest neighbors of $\mathbf{x}$ to get a smooth geodesic distance over the whole space

$$\|\mathbf{x}-\mathbf{p}\|_{\text{geo(k)}} = \min_{p^* \in \mathcal{P}_k(\mathbf{x})} \left\{ \|\mathbf{x}-\mathbf{p}^*\| + d(\mathbf{p}^*, \mathbf{p}) \right\} \qquad (7)$$

where $\mathcal{P}_k(\mathbf{x})$ denotes the set of the k-nearest neighbors in the corresponding proximity graph.

Alternatively, this problem could possibly be solved by utilizing natural coordinates [BC00]. However, the computational costs seem to be very high (albeit a constant factor over insertion of a point in the Delaunay graph).

5. Results

We have implemented our new point cloud surface definition in C++. It is easy to implement and can be evaluated very fast: once the graphs are built, we can evaluate $f(\mathbf{x})$ simply by finding a nearest neighbor, traversing the graph, computing a number of weights from the CPSP map, and finally one eigenvector by Cholesky decomposition.

First of all, Figure 7 shows the performance that can be achieved using our new surface definition for a reasonable choice of h. Although our implementation is not fully optimized, the performance is of the same order as that of the Euclidean kernel.

Figure 10 illustrates the quality depending on the Euclidean kernel and our new geodesic one, respectively. Moreover, in order to give a numerical hint for the quality, we determined the root mean square error (RMSE) for the deviation (i.e., distance) of the reconstructed surface from the original surface. Obviously, our geodesic kernel approximates the surface very well, while the Euclidean kernel produces several artifacts. Even when the bandwidth h (see Equation 3) is chosen optimally with respect to the RMSE,

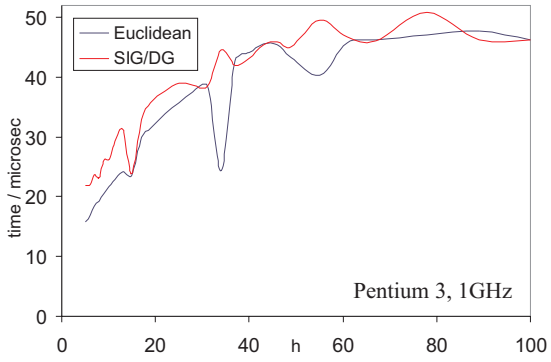


Figure 7: *Average evaluation time of* $f(\mathbf{x})$ *depending on the kernel bandwidth h (size of point cloud: >2000 points). The timings for* $\text{SIG}(\mathcal{P})$ *and* $\text{DG}(\mathcal{P})$ *are nearly identical (therefore, we omit one curve). Please note that our implementation is not yet fully optimized.*

the Euclidean kernel produces severe artifacts (see Figure 10e).

We also performed experiments to assess the sensitivity of our surface definition with respect to the kernel bandwidth h. The plots in Figure 8 (left and center) show for two different example surfaces that our new kernel is less sensitive towards the choice of h than the old one: for a large range of the bandwidth, the RMSE using our new surface definition is quite low. In contrast, only for a small bandwidth the Euclidean kernel yields a relatively low RMSE. Note that in almost every case the RMSE of the Euclidean kernel is larger than the RMSE of our new kernel. Note further, that the minimal RMSE of our new definition is clearly smaller than the minimal RMSE of the old one.

It might seem that there are still two parameters in our new approach, which require fine-tuning: r, the radius for our

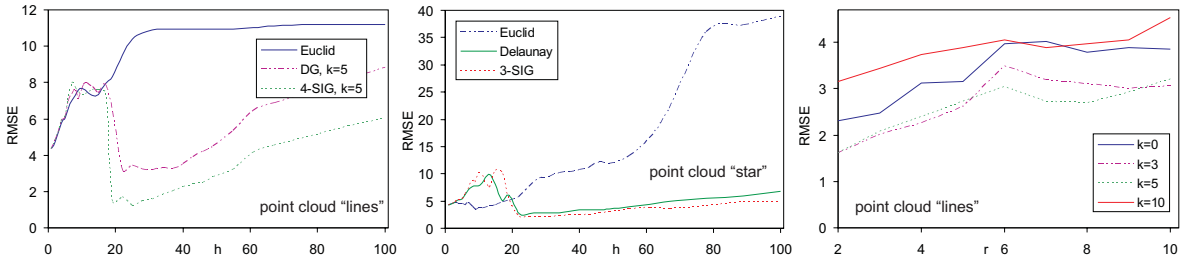


Figure 8: *Left and center: RMSE depending on the bandwidth, h, of the kernel. Our new kernel is less sensitive towards the choice of h than the old one. Right: RMSE for* r-SIG($\mathcal{P}$) *depending on different r.*

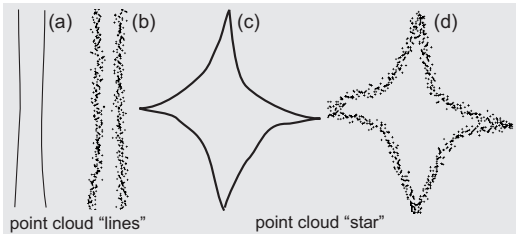


Figure 9: *Original (not reconstructed) surfaces (a,c) from which the noisy point clouds (b,d) have been generated that are used for the evaluation in Fig. 8.*

modified sphere-of-influence graph r-SIG, and the parameter k in our geodesic approximation $\|\cdots\|_{\text{geo}(k)}$. However, numerous measurements for different point clouds showed, that for $k, r \in [3 \ldots 6]$, these two parameters are very robust and yield very similar results. Figure 8 (right) shows the RMSE depending on both parameters.

6. Conclusion and Future Work

We have presented a new surface definition that utilizes proximity graphs, k-d trees, and moving least squares to approximate geodesic distances.

Overall, our new surface definition yields implicit functions over point clouds, the zero sets of which are much closer to the original surface than the simple moving least squares approach. At the same time, our definition can be evaluated quite fast. In addition, the auxiliary data structures can be constructed efficiently and incur only little additional storage.

Of course, our method can be utilized for other variants of point cloud surfaces as well, such as local polynomial approximations (which build on top of moving least squares).

In rare cases, our r-SIG tends to have a few more edges than necessary, which can result in unintentional shortcuts, e.g, in cavities. Together with our non-Euclidean distance computation, farther points (according to the true geodesic distance) could be weighted more than nearer ones. This should be examined in the future.

Our approach is well-suited for static settings, where the graphs and the CPSP maps can be precomputed. It would be desirable to adapt our approach to deformable point clouds as well.

Finally, we will investigate methods to adjust the kernel bandwidth automatically and locally by utilizing the proximity graph.

Acknowledgements

We would like to extend our thanks to our anonymous reviewers for their constructive and thorough comments.

This work is partially supported by DFG grant DA155/29-1 "Benutzerunterstützte Analyse von Materialflusssimulationen in virtuellen Umgebungen" (BAMSI), and the DFG program "Aktionsplan Informatik" by grant ZA292/1-1.

References

[AA03] ADAMSON A., ALEXA M.: Approximating and intersecting surfaces from points. In *Proc. Eurographics Symp. on Geometry Processing* (2003), pp. 230–239.

[AA04] ADAMSON A., ALEXA M.: Approximating bounded, non-orientable surfaces from points. In *Shape Modeling International* (2004). to appear.

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics 9*, 1 (2003), 3–15.

[AK04] AMENTA N., KIL Y.: Defining point-set surfaces. In *Proc. of SIGGRAPH* (2004). to appear.

[AMN*98] ARYA S., MOUNT D. M., NETANYAHU N. S., SILVERMAN R., WU A.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM 45* (1998), 891–923.

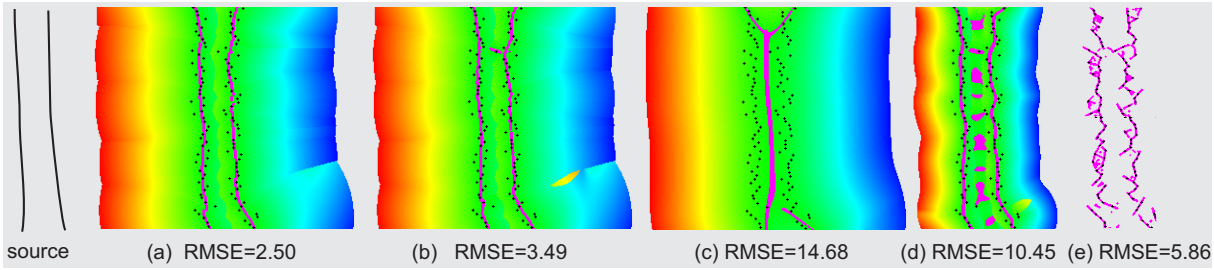


Figure 10: *Root mean square error (RMSE) for a noisy point cloud (left: original surface). (a)* DG($\mathcal{P}$) *with edges larger than second quartile are pruned, (b)* 2-SIG($\mathcal{P}$)*, (c) Euclidean distance kernel, (d) same with reduced bandwidth h, (e) Euclidean distance kernel with optimal bandwidth h that yielded the minimum RMSE (notice the inferior surface quality).*

[ASCL02] AMENTA N., S. CHOI T. K. D., LEEKHA N.: A simple algorithm for homeomorphic surface reconstruction. *Intl. Journal on Computational Geometry & Applications 12* (2002), 125–141.

[BC00] BOISSONNAT J.-D., CAZALS F.: Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proc. 16th Annual Symp. on Computational Geometry* (2000), ACM Press, pp. 223–232.

[BLS00] BOYER E. D., LISTER L., SHADER B.: Sphere-of-influence graphs using the sup-norm. *Mathematical and Computer Modelling 32* (2000), 1071–1082.

[BWG03] BALA K., WALTER B., GREENBERG D. P.: Combining edges and points for interactive high-quality rendering. *Proc. of SIGGRAPH* (2003), 631–640.

[CH90] CHEN J., HAN Y.: Shortest paths on a polyhedron. In *Proc. 6th ACM Symp. on Computatinal Geometry* (1990), pp. 360 – 369.

[Dev02] DEVILLERS O.: The Delaunay hierarchy. *Internat. J. Found. Comput. Sci. 13* (2002), 163–180.

[DG03] DEY T. K., GOSWAMI S.: Tight cocone: A water tight surface reconstructor. In *Proc. 8th ACM Sympos. Solid Modeling Appl.* (2003), pp. 127–134.

[DG04] DEY T. K., GOSWAMI S.: Provable surface reconstruction from noisy samples. In *Proc. Symp. on Computational Geometry* (2004). to appear.

[Dwy95] DWYER R. A.: The expected size of the sphere-of-influence graph. *Computational Geometry: Theory and Applications 5*, 3 (Oct. 1995), 155–164.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *Proc. of SIGGRAPH* (1992), pp. 71–78.

[HUHJ01] HECKEL B., UVA A. E., HAMANN B., JOY K. I.: Surface reconstruction using adaptive clustering methods. In *Computing Supplement*, vol. 14. 2001, pp. 199–218. Dagstuhl 1999.

[JT92] JAROMCZYK J. W., TOUSSAINT G. T.: Relative neighborhood graphs and their relatives. In *Proc. of the IEEE* (1992), vol. 80, pp. 1502–1571.

[KS00] KANAI T., SUZUKI H.: Approximate shortest path on a polyhedral surface based on selective refinement of the discrete graph and its applications. In *Proc. Geometric and Processing* (2000), pp. 241 – 250.

[KZ04] KLEIN J., ZACHMANN G.: Point cloud collision detection. In *Computer Graphics Forum (Proc. EUROGRAPHICS)* (2004). to appear.

[Lee00] LEE I.-K.: Curve reconstruction from unorganized points. *Computer Aided Geometric Design 17*, 2 (2000), 161–177.

[Lev03] LEVIN D.: Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, Brunnett H., Mueller, (Eds.). Springer, 2003.

[MQ03] MICHAEL T. S., QUINT T.: Sphere of influence graphs and the l_∞-metric. *Discrete Applied Mathematics 127*, 3 (2003), 447 – 460.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. In *Proc. of SIGGRAPH* (2003), pp. 463–470.

[PvBZG00] PFISTER H., VAN BAAR J., ZWICKER M., GROSS M.: Surfels: Surface elements as rendering primitives. In *Proc. of SIGGRAPH* (2000), pp. 335–342.

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proc. of SIGGRAPH* (2000), pp. 343–352.

[Rog63] ROGERS C.: Covering a sphere with spheres. *Mathematika 10* (1963), 157–164.

[VB94] V. BARNETT T. L.: *Outliers in Statistical Data*. John Wiley and Sons, New York, 1994.

[Vel93] VELTKAMP R. C.: 3D computational morphology. *Computer Graphics Forum (Proc. EUROGRAPHICS)* (1993), 115–127.

[ZPvBG02] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: EWA splatting. *IEEE Trans. on Visualization and Computer Graphics 8*, 3 (2002), 223–238.