Smoothing Noisy Point Clouds with Delaunay Preprocessing and MLS

Tamal K. Dey* Samrat Goswami† Jian Sun‡

The Ohio State University

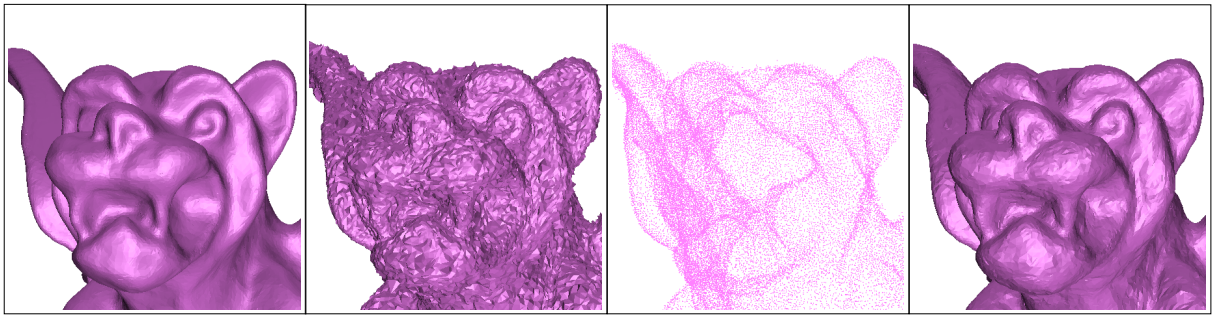

Figure A: Effectiveness of the smoothing technique: The vertices of a smooth GARGOYLE(leftmost) are perturbed with a Gaussian noise (middle-left). These perturbed vertices are taken as input point cloud data (middle-right). The result after our MLS smoothing with Delaunay preprocessing (rightmost). All pictures are flat shaded in this paper.

Abstract

The Moving Least Squares (MLS) method has been very effective in smoothing point cloud data that may be noisy. However, the quality of this smoothing depends on how well one can estimate the neighborhood of the points on the sampled surface. In this work we present a Delaunay based scheme that estimates this neighborhood from point clouds possibly corrupted with noise. This neighborhood information is used for designing a very simple kernel function for the MLS which produces quality approximation and smoothing.

CR Categories: [Computer Graphics]: Modeling—Point-based Graphics, Surface Reconstruction.

Keywords: smoothing, point cloud, meshing.

1 Introduction

Shape modeling from their point samples has become an effective paradigm because of its light-weight input specification and the flexibility offered by the process. The recent work on surface reconstruction [3, 4, 6, 11, 16, 20], medial axis approximation [5, 9], solid modeling [15], rendering [1] are only a few examples attesting this fact. The point sample, often acquired by some scanning process, contains noise. The Moving Least Squares (MLS) pioneered by Levin [17, 18] has been proposed to deal with such noisy samples by Alexa et al. [2]. Following this work, some other results were obtained by this powerful technique including the work of Alexa et al. [1], Pauly et al. [15] and Pauly and Gross [14].

Given a point sample P from a smooth surface $\Sigma \in \mathbb{R}^3$, The MLS applied to a point $p \in P$ moves it to a smooth surface M that approximates Σ. The quality of this approximation depends upon how well the feature sizes of Σ are approximated from P. The influence of other points on p is captured by a weight function called the *kernel* for p. Ideally, only points from a neighborhood of p which is smaller than the local feature sizes of Σ should have large weights. In fact, for reducing computation time only these points can be allowed to influence the movement of p. All previous works [2, 14, 15] use k-nearest neighbors for this purpose. However, determining appropriate k is not always easy.

In this paper we propose a simple kernel function that uses Delaunay triangulations of the point set. The combination of the MLS with this Delaunay kernel achieves quality smoothing and approximation. The point cloud is processed to extract a subset of points and a a preliminary surface is computed interpolating them. This surface may have undesirable undulations due to noise. Nevertheless, the neighborhood of a point determined by the Delaunay triangles on this surface does not exceed the local feature size under a reasonable noise model. This neighborhood is used to determine the kernel function for a subsequent smoothing step.

2 Moving least squares

Given a point set $P \in \mathbb{R}^3$, the MLS surface M is defined as the stationary set of a projector operator $\Phi(\cdot)$ where $x \in \mathbb{R}^3$ is projected by Φ to $\Phi(P,x) \in M$. The projection $\Phi(P,x)$ is computed as follows. First, a plane

$$H = \{x \in \mathbb{R}^3 \,|\, x \cdot n - D = 0\} \tag{1}$$

*email: tamaldey@cis.ohio-state.edu
†email: goswami@cis.ohio-state.edu
‡email: sunjia@cis.ohio-state.edu

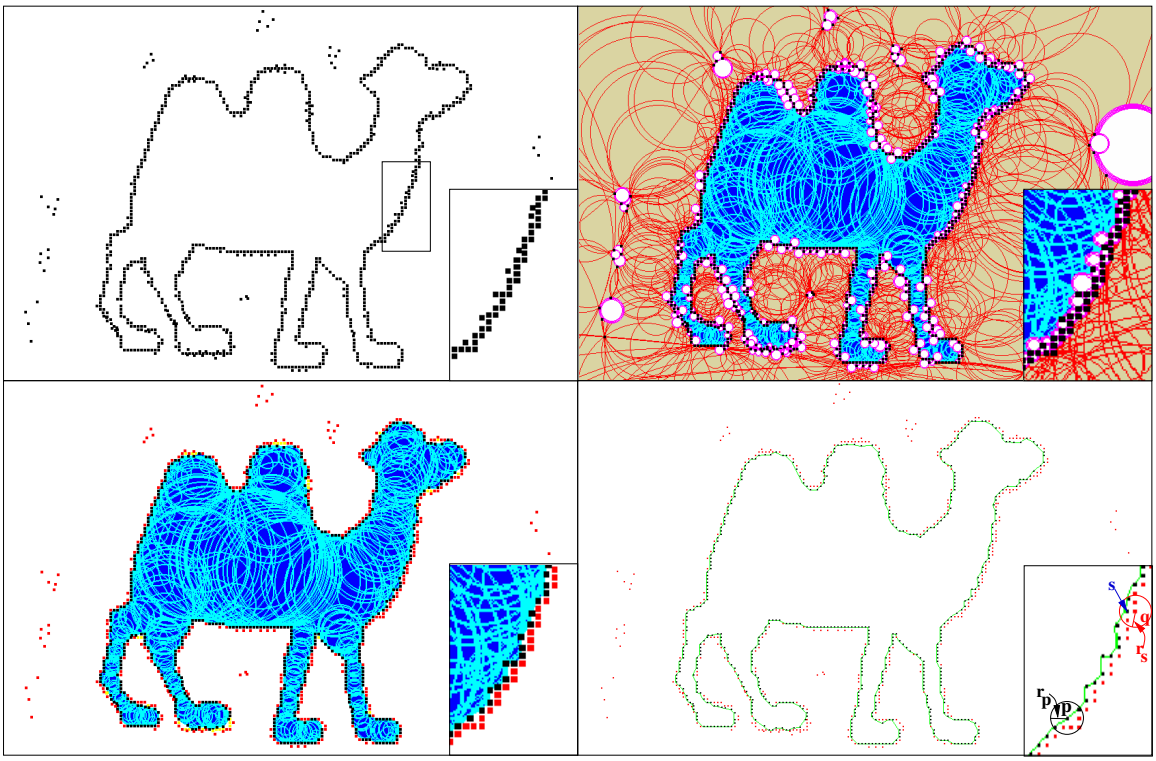

Figure 1: Point processing: A noisy point sample along with outliers (top-left), small Delaunay balls are shaded white (top-right), inner big Delaunay balls selected (bottom-left), reconstructed shape from the points on these balls (bottom-right).

is determined by minimizing the weighted sum of the squared distances

$$\sum_{p \in P} (p \cdot n - D)^2 \kappa(\|p - q\|), \tag{2}$$

where q is the projection of x onto H and κ is a smooth function called the MLS kernel. After transforming all points into a local coordinate system defined on H, a second least square optimization is carried out. This optimization gives a polynomial $g(u,v)$ of a chosen degree which locally approximates M. The projection of x onto M is given by $\Phi(P,x) = q + g(0,0) \cdot n$.

Usually a Guassian $\kappa(t) = \exp(-t^2/\sigma^2)$ is used where σ is a parameter that determines the local feature sizes of M. The local feature at a point x on a surface can be defined as the distance of x to the medial axis of the surface [3]. Ideally σ should be adaptive to the local feature sizes of Σ from where P is sampled, or at least should not exceed the local feature sizes of Σ.

Our main contribution is to show that we can choose a very simple $\kappa(\cdot)$ which respects the local feature sizes under a reasonable noise model. The noise model assumes that P is perturbed from Σ within some fraction of the local feature sizes. It forms a dense sample of Σ when projected onto Σ. Notice that these two conditions are reasonable as otherwise the perturbation can be arbitrarily large creating a point cloud without any resemblance to Σ, or can be arbitrarily sparse without capturing the features of Σ.

We employ a point processing step from [8] to reconstruct Σ from P. If P is noisy, this surface may have noisy undulations. Each point p on this surface has a neighborhood size proportional to the sampling density. This neighborhood is taken for designing the kernel κ of the MLS.

3 Point processing

If P is noise-free, one could reconstruct a surface interpolating P using any of the recent surface reconstruction algorithms and then could take the appropriate neighborhood of the points for the kernels [3, 4, 6]. In case P is noisy, this proposition fails unless an algorithm is designed to reconstruct a surface from noisy sample points. We use the algorithm of Dey and Goswami [8] for this purpose.

The Delaunay triangulation of the input point set P is denoted $\mathrm{Del}\,P$. Its dual, the Voronoi diagram, is denoted $\mathrm{Vor}\,P$ and a Voronoi cell for a point $p \in P$ is denoted V_p. The circumscribing balls of the Delaunay tetrahedra are called *Delaunay balls*.

To understand the rationale behind the method, first assume that P is noise-free. For a point $p \in P$, the inner (outer) pole of p is the Voronoi vertex that lies inside (outside respectively) Σ and is farthest from p among all other vertices of V_p. The Delaunay balls centering the poles are called *polar balls*. It is known by a result of Amenta, Choi and Kolluri [5] that the polar balls centering the inner (outer) poles approximate the shape bounded by Σ (unbounded component of $\mathbb{R}^3 - \Sigma$ respectively). It is easy to compute the poles from the Voronoi diagram $\mathrm{Vor}\,P$. However, in absence of Σ, one needs a mechanism to separate the inner poles from the outer ones in order to obtain an approximation of Σ. The polar balls satisfy the following intersection depth property. Two inner (outer) polar balls that cricumscribe adjacent tetrahedra intersect deeply while an inner and a outer polar ball intersect only in a shallow manner. The depth of intersection is measured by the angle at which the boundaries of the two balls intersect. A breadth-first search starting from an unbounded polar ball, which is outer for sure, can collect all of them if the walk moves from an outer polar ball to an adjacent polar ball only if they intersect deeply.

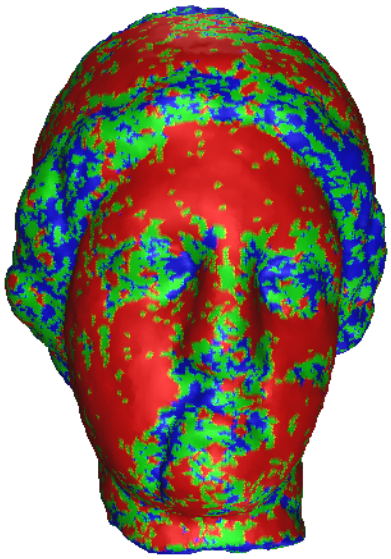

Figure 2: Spread radii of the points indicated with colors. Highest one-thirds are colored red, middle one-third are colored green, and lowest one-third are colored blue.

The above algorithm does not work when P is noisy. A main reason for this is that the intersections between polar balls do not follow the intersection depth property. We circumvent this difficulty by observing that, even with noisy point sample where the points are not perturbed more than a small fraction of the local feature sizes, some of the Delaunay balls behave like polar balls as in the noise-free case. We identify them by their relative sizes compared to the nearest neighbor distances. Let B be a Delaunay ball cricumscribing four points $p_i, i = 1,..,4$. We say B is *big* if the radius of B is more than $\rho > 0$ times bigger than the nearest neighbor distance of any p_i where ρ is an input parameter. In practice we choose $\rho = 1.5$ and instead of taking the nearest neighbor distance, we take the average distance of the 5 nearest neighbors. After identifying the big Delaunay balls, we partition them using the intersection depth property as in the noise-free case. It is proved in [8] that, under the noise model as described before, big Delaunay balls satisfy the intersection depth property and the union of inner big Delaunay balls approximate Σ. In particular, one can reconstruct Σ from the subset of P that lie on the boundary of the inner big Delaunay balls. Figure 1 shows the steps of this algorithm. The surface $\hat{\Sigma}$ which is output by this algorithm is used to determine the kernel for MLS.

Outliers. The point processing step is quite robust against outliers. Some of the Delaunay balls incident to outliers are detected small since their average 5-nearest neighbor distances are relatively big. The other big Delaunay balls intersect other outer balls deeply and hence are detected outer. As a result, the outliers do not lie on inner big Delaunay balls which are used for preliminary surface. Figure 1 illustrates this effect.

4 Delaunay kernel

For each point $p \in P$ we determine the kernel function as follows. In the sum of equations 1 and 2, all points of P are considered. This leads to a prohibitive computation for moderately large data sets. To circumvent this problem, usually only a few neighbors such as

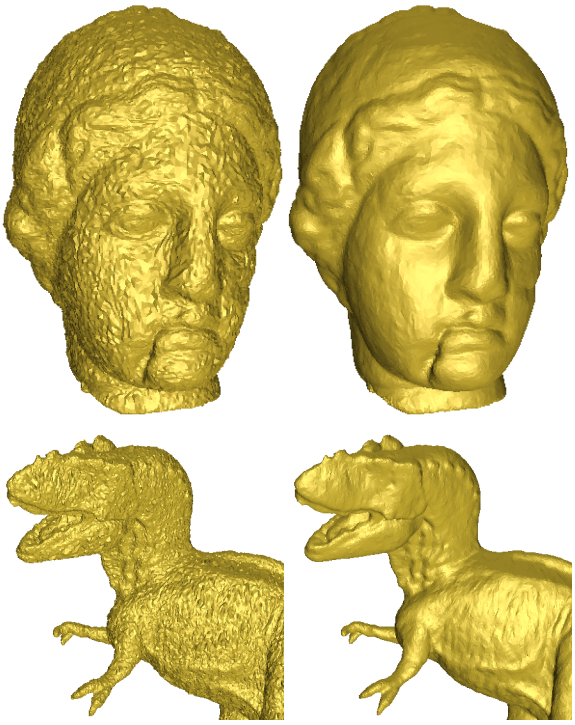

Figure 3: The preliminary surfaces after the point processing step (left). The surfaces with the same vertex connectivity after the vertices are smoothed by the MLS with the Delaunay kernels (right).

k-nearest neighbors of p for a pre-assigned number k are taken in these sums. Let the set of points of P that are considered while computing the MLS position of a point be called its *spread*. We propose a natural, adaptive spread for each $p \in P$ and the corresponding kernel.

The distance between a point p on $\hat{\Sigma}$ and its neighbors in $\hat{\Sigma}$ is less than the local feature size of p in Σ. Using this rationale, first we eliminate some of the points from P as follows. Let Q be the vertex set of $\hat{\Sigma}$. We collect a subset $P' \subseteq P$ that lie close to the set Q. Let r_p be the distance between a vertex p of $\hat{\Sigma}$ and the farthest vertex connected to it on $\hat{\Sigma}$. A point is in P' if it lies within r_p distance of a point $p \in Q$. Observe that outliers are eliminated from P' as they lie far away from all points of $\hat{\Sigma}$. The set P' is smoothed by MLS. We determine the spreads for each point in P' as follows.

If $p \in Q$, the neighbors of p on $\hat{\Sigma}$ are taken as its spread. If p is not a vertex in $\hat{\Sigma}$, we take s, the nearest vertex of p on $\hat{\Sigma}$. The spread of p is the set of points lying within a distance of r_s. The spreads for two points, one on the surface $\hat{\Sigma}$ and one not on $\hat{\Sigma}$, are indicated in the bottom-right picture of Figure 1. The radii of the spreads as computed by our method are shown in Figure 2. Regions with relatively small features have small radii. This helps in setting the kernel width smaller than the local feature sizes.

Let N_p denote the spread of p as computed above. The kernel $\kappa(p,q)$ of p can be taken $\exp^{(-\|p-q\|^2)/r^2}$ where $q \in N_p$ and r is the radius of the spread. Instead we simplify the kernel as

$$\begin{aligned} \kappa(p,q) &= 1 \text{ if } q \in N_p, \\ &= 0 \text{ otherwise.} \end{aligned}$$

We call this kernel the *Delaunay kernel*. The results with this simplified kernels are shown in Figure 3 and Figure 4.

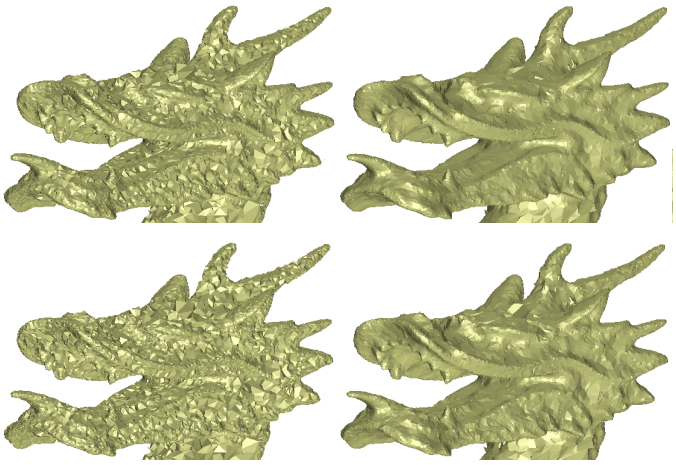

Figure 4: DRAGON with different noise levels smoothed by our method. The top row contains a Guassian noise with a multiplicative factor of 1000, and the bottom one with a multiplicative factor of 3000.

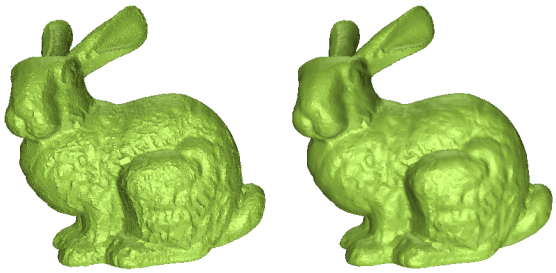

Figure 5: The surface reconstructed out of the noisy Stanford Bunny sample (left), the surface after smoothing the vertices with the Delaunay kernel (right).

5 Results and comparisons

We implemented our algorithm with the CGAL library. This library provides fast, robust Delaunay triangulations. For example, for GARGOYLE with approximately 100,000 points, the entire smoothing including the point processing step takes 137 seconds on a 2.8 GHZ pentium 4 maching with 1 GB RAM. Table 6 shows the times for different models.

object	# points	Time (sec.)
VENUS	50K	52.78
GARGOYLE	100K	134.32
DRAGON	100K	134.35
TYRA	100K	137.13
BUNNY	362K	516.36

Figure 6: Time data.

We compared our method with the MLS method implemented in POINTSHOP3D [20]. The MLS kernel in this method estimates the feature sizes using k-nearest neighbors. It suffers from two difficulties. First, in thin regions where usually undersampling occurs, k-nearest neighbors can reach beyond feature sizes creating severe distortion in features as Figure 7 shows. Secondly, it cannot handle outliers. The noise on the bottom of the GARGOYLE creates some outliers that MLS does not re-position. Our method processes them gracefully, see Figure 8.

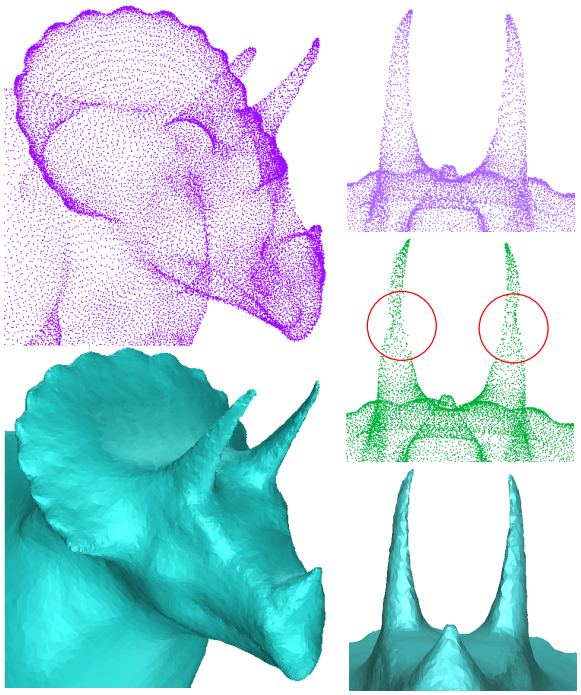

Figure 7: Noisy point cloud of TRICERATOPS (upper left) smoothed by our method (lower left). MLS with k-nearest neighbor kernels (upper middle) distorts horns, Delaunay kernel preserves them (lower right).

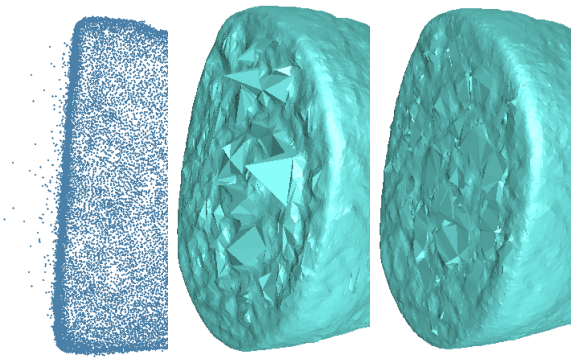

Figure 8: MLS with k-nearest neighbor kernels does not re-position some of the outliers. The vertices of the GARGOYLE was perturbed to create some outliers at the bottom. The surface with the original connectivity after MLS smoothing shows the outliers at the bottom of GARGOYLE (middle). MLS with Delaunay kernel removes the outliers (right).

There are a number of techniques available for smoothing meshes [7, 12, 13, 19]. Although we are more concerned with smoothing point cloud data, we chose the bilateral mesh denois-

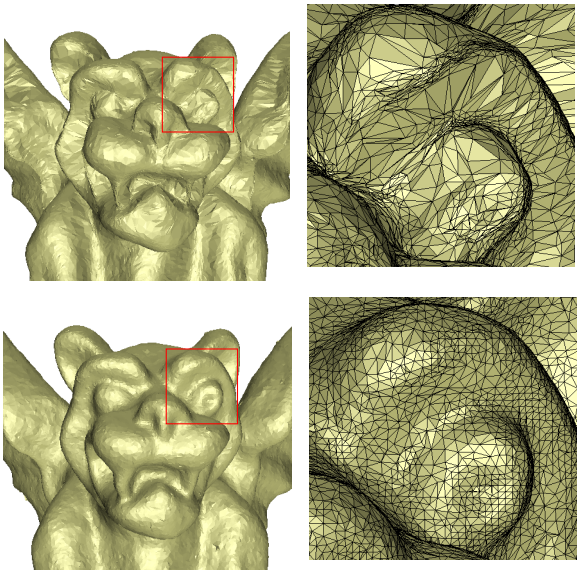

Figure 9: Noisy mesh of GARGOYLE after bi-lateral denoising (top), the mesh after reconstructing with the vertices smoothed by our method (bottom).

ing [10] for a comparison of the mesh qualities. We took the mesh of GARGOYLE and perturbed it with a Gaussian noise. The outputs of the bilateral denoising moves the vertices towards sharp features and tend to produce an uneven mesh. We smoothed the vertices first with our method and then reconstructed the surface out of the smooth point sample. The two surfaces with highlights are shown in Figure 9.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE TVCG*, to appear.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. T. Silva. Point set surfaces. *Proc. IEEE Visualization* (2001), 21–28.

[3] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discr. Comput. Geom.* **22** (1999), 481–504.

[4] N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Internat. J. Comput. Geom. Applications* **12** (2002), 125–141.

[5] N. Amenta, S. Choi and R. Kolluri. The power crust, union of balls, and the medial axis transform. *Comput. Geom. Theory Appl.* **19** (2001), 127–153.

[6] J.-D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbor interpolation of distance functions. *Proc. 16th Annu. Sympos. Comput. Geom.* (2004), 223–232.

[7] M. Desbrun, M. Meyer, P. Schröder and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proc. SIGGRAPH 99* (1999), 317–324.

[8] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. *Proc. 20th Annu. Sympos. Comput. Geom.* (2004), to appear.

[9] T. K. Dey and W. Zhao. Approximating the medial axis from the Voronoi diagram with a convergence guarantee. *Algorithmica* **38** (2004).

[10] S. Fleishman, I. Drori and D. Cohen-Or. Bilateral mesh denoising. *SIGGRAPH 03* (2003), 950–953.

[11] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. Surface reconstruction from unorganized points. *Proc. SIGGRAPH 92* (1992), 71–78.

[12] T. R. Jones, F. Durand and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *Proc. SIGGRAPH 2003* (2003), 943–949.

[13] Y. Ohtake, A. Belyave and H.-P. Seidel. Mesh smoothing by adaptive and anisotropic Gaussian filter. *Vision, Modeling and Visualization* (2002), 203–210.

[14] M. Pauly, M. H. Gross and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. *Proc. IEEE Visualization* (2002), 163–170.

[15] M. Pauly, R. Keiser, L. P. Kobbelt and M. H. Gross. Shape modeling with point-sampled geometry. *Proc. SIGGRAPH 2003* (2003), 641–650.

[16] M. Pauly and M. H. Gross. Spectral processing of point-sampled geometry. *Proc. SIGGRAPH 01* (2001).

[17] D. Levin. Mesh-independent surface interpolation. *Advances in Comput. Math.*

[18] D. Levin. The approximation power of moving least-squares. *Math. Comp.* **67** No. 224.

[19] G. Taubin. A signal processing approach to fair surface design. *Proc. SIGGRAPH 95* (1995), 351–358.

[20] M. Zwicker, M. Pauly, O. Knoll and M. Gross. Pointshop 3D: An interactive system for point-based surface editing. *ACM Trans. Graphics* **21** (2002), 322-329.