Fairing of Point Based Surfaces

U. Clarenz[1], M. Rumpf[1], A. Telea[2]
[1] Institut für Mathematik, University of Duisburg-Essen, Germany
{*clarenz | rumpf*}*@math-uni-duisburg.de*
[2] Department of Mathematics and Computer Science
Eindhoven University of Technology, the Netherlands
alext@win.tue.nl

Abstract

We present a framework for processing point-based surfaces via partial differential equations (PDEs). Our framework allows an efficient and effective way to bring well-established PDE-based surface processing techniques to the field of point-based representations. We demonstrate the method by a PDE-based surface fairing application.

1. Introduction

Surface processing tools and techniques are widespread in computer graphics, animation, medical imaging, computer aided modelling, and computer vision. An important class of surface processing operations can be described via partial differential equations (PDEs). Traditionally, for the discretization of PDEs, the triangle strip surface representation is used. Recently, a number of point based representations have been proposed as an alternative to triangles for 3D surfaces. Point based surfaces have a number of important advantages when compared to their triangular counterparts. First, no 'mesh', or connectivity information has to be stored explicitly. This allows a simple and compact representation, ideal for fast rendering and editing. When combined with advanced rendering techniques such as splatting [17, 14], point based surfaces outperform triangle meshes in terms of rendering quality and data storage flexibility.

Processing point-based surfaces via PDEs should add the modelling power of PDE representation to the flexibility of the point based model. However, defining and solving PDEs using finite elements on point based surfaces is not straightforward. The main problem in defining and solving PDEs is that point-based surfaces are mesh-less, so there is no direct way to define classical finite elements on those "surfaces". One way to obtain a Lipschitz surface is the application of triangulation algorithms [11, 10, 4, 13]. However, we shall not consider this option here, as it basically undermines the fundamental philosophy and advantages of a point based model. Furthermore such techniques may produce surfaces lacking the desired amount of smoothness, as described in [2, 1]. Smoothness can be gained by applying smoothing operations to the obtained triangle mesh, such as iterative Laplacian smoothing [15], curvature flow fairing [9, 6], or discrete variational fairing [12]. Alternatively, increased smoothness can be obtained by using higher local approximations, such as piecewise polynomials [16] or radial basis functions [14, 17].

The main contribution of this paper is an approach to finite element based PDEs on point surfaces. We proceed by constructing a number of local finite element matrices that represent the point set surface properties over small point neighborhoods. These matrices are next assembled in a single matrix which allows the PDE discretization and solving on the complete surface. We illustrate our approach by a well-known surface fairing algorithm based on mean curvature flow.

2. PDEs on point clouds

The problem of applying PDE-techniques onto point clouds consists in the lack of a sensible function space as e.g. C^0 or H^1. Therefore it is not straightforward to generalize classical concepts of finite elements to point clouds.

In our approach, we proceed by constructing a local tangent space and consider the local projection of the point set onto this tangent space. We are then able to define stable coupling quantities between neighbor points, using a strictly local Delaunay meshing. The

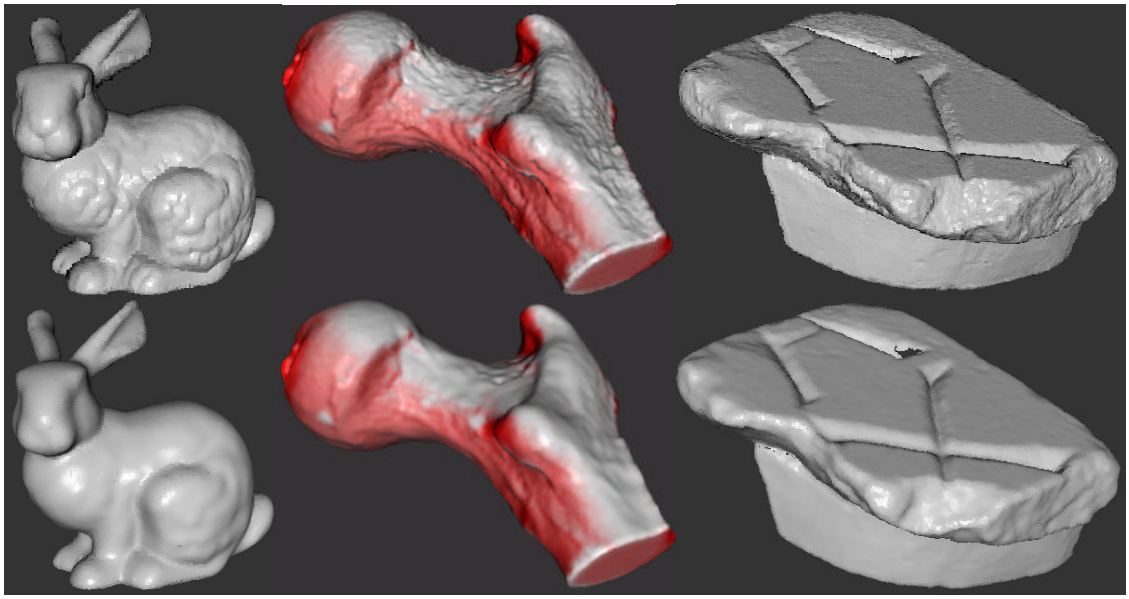

Figure 1. Top row: initial surfaces. Bottom row: surfaces faired via diffusion

mentioned coupling quantities are suitable discretizations of the off-diagonal entries in the global stiffness matrix we aim to recover. The local tangent spaces of different points usually do not coincide, which induces a loss of symmetry in our matrix. To remove this problem, we finalize the matrix construction by applying a suitable symmetrization. The diagonal entries of the stiffness matrix are next defined based on a requested invariance property with respect to constant functions. We proceed similarly for simpler cases of the mass matrix and the right hand side of the considered PDE. The complete approach is detailed in [7].

3. Point cloud classification

We proceed by defining, for every point x in the considered point cloud classification measures which especially define tangent spaces and local smoothness. To this aim we take into account the points n_i in a small neighborhood N of x. The size of N should be chosen such that a) it contains a minimal number of points for stable computation of a tangent space and b) the radius of N is not larger than the size of the features we want to be visible in the approximation. For the first requirement, N can be efficiently computed using the k nearest neighbors of x, for given k. For the second requirement, N can be defined as the ball $B_\epsilon(x)$ of given radius ϵ centered at x. In practice, combinations of the two criteria give the best results. We prescribe a minimum number of neighbors k_{min}, to enforce the first requirement. If the k_{min}^{th} closest neighbor of x is closer than the prescribed minimal feature size ϵ, we consider all additional nearest neighbors in $B_\epsilon(x)$.

For the classification, we use the *zero and first order moments* of N. Moments have several proven properties that allow us to robustly compute the tangent spaces as well as to distinguish between smooth and non-smooth surface parts, both as a function of the ball size ϵ, see [8]. Robustness is clearly needed in the tangent plane computation. Distinguishing smooth from non-smooth surface areas is needed for our surface fairing application, detailed in Sec. 4.

For a continuous surface $\mathcal{M}$, the zero moment is given by the local barycenter of $\mathcal{M}$ with respect to a Euclidian ball $B_\epsilon(x)$ centered at x:

$$M_\epsilon^0(x) := \fint_{B_\epsilon \cap \mathcal{M}} x \, \mathrm{d}x \, .$$

The first order moment is then defined as as:

$$M_\epsilon^1(x) \quad := \quad \fint_{B_\epsilon \cap \mathcal{M}} (x - M_\epsilon^0(x)) \otimes (x - M_\epsilon^0(x))) \, \mathrm{d}x$$

where $y \otimes z := (y_i z_j)_{i,j=1,\ldots,3}$.

Finally, we define the *zero moment shift* as

$$N_\epsilon(x) = M_\epsilon^0(x) - x \, .$$

We can use these moment information to distinguish between smooth surface domains and non-smooth areas such as close to edges and corners by a scaling anal-

ysis. For the scaling properties of the moments we refer to [8].

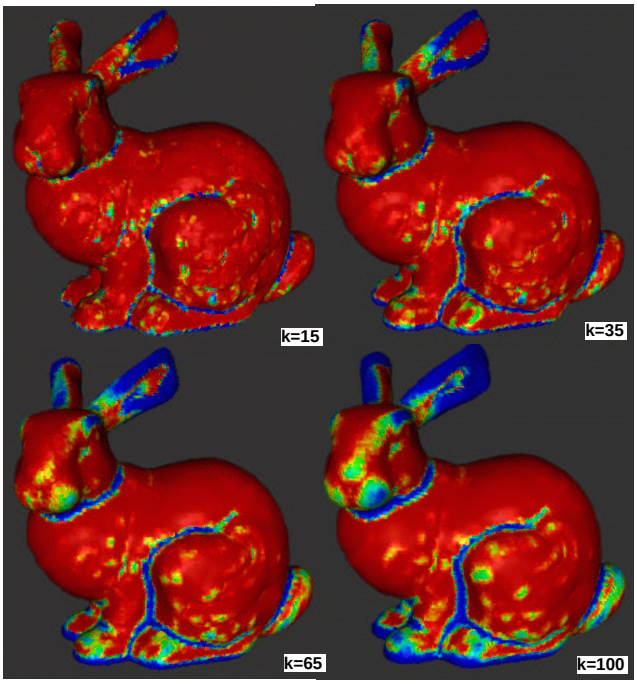

Figure 2. Classifier for different k-closest point values

Consequently, we can use moments to construct a *surface classifier* that quantitatively indicates the presence of such features. We define the classifier $\mathcal{C}_\epsilon$ as

$$\mathcal{C}_\epsilon = G\left(\frac{||N_\epsilon||\lambda_{min}}{\epsilon\lambda_{max}}\right)$$

where $G(s) = \frac{1}{\alpha+\beta s^2}$ with suitably chosen α, $\beta > 0$. λ_{min} and λ_{max} are the smallest, respectively largest eigenvalue of the first order moment. The reason for the above definition is twofold. First, the quantity $\frac{||N_\epsilon||}{\epsilon}$ is of order ϵ on smooth areas and 1 close to singularities. Secondly, close to edges, the quantity $\frac{\lambda_{min}}{\lambda_{max}}$ is proportional to $\cos^2\varphi$ where φ is the edge apex angle. Combining the above two, we obtain a quantity $\mathcal{C}_\epsilon$ that robustly detects the presence of edges. In our applications, we have fixed $\alpha = 0.01$ and $\beta = 100$. The function G ensures that $\mathcal{C}_\epsilon$ is much larger in smooth areas than close to edges, making the classification easier. Figure 2 shows the classifier $\mathcal{C}_\epsilon$ for the bunny model for four different values of ϵ. Specifically, ϵ was implicitly determined by choosing the first k closest points to a given point. Blue indicates low values (edges), whereas red shows high values (smooth areas). Different 'features' are obviously detected at different scales.

Our classifier is roughly similar with the so-called surface variation introduced by Pauly [14], which is defined as the ratio of the smallest eigenvalue to the sum of eigenvalues of a covariance matrix similar, up to scaling, to our first order moment. However, we combine in our classification information from *both* zero and first order moments.

Let us briefly explain, how we use the eigenvectors of the first moment M^1_ϵ to define a local tangent plane. If $\lambda_0 > \lambda_1 > \lambda_2$ are the eigenvalues of the 3 by 3 symmetric matrix M^1_ϵ, then the corresponding eigenvector e_2 is the plane normal, whereas e_1 and e_0 form a 2D coordinate system in the plane itself. Our tangent plane computation is similar to the principal component analysis used in [2, 14, 16]. However, as explained already, we prefer our moment-based approach as it comes with proven scaling properties with respect to the ball size ϵ. Concisely, ϵ plays the role of a filter size: the tangent plane corresponds to a surface containing only features larger than ϵ.

For details on the neighborhood computation on the tangent plane see also [5, 10, 3].

4. Fairing of Point Based Surfaces

As an application of our framework for PDEs on point based surfaces, we consider surface fairing using anisotropic geometric diffusion. In this type of application, surface geometrical noise is smoothed out, whereas features such as edges are preserved or possibly even enhanced [6, 9]. This is especially useful for point surfaces acquired via a possibly noisy laser scanning process. More precisely, given an initial compact embedded manifold $\mathcal{M}_0$ in $\mathbb{R}^3$, we compute a family of faired manifolds $\{\mathcal{M}(t)\}_{t\in\mathbb{R}^+_0}$, with corresponding coordinate mappings $x(t)$. The time t describes the fairing process and $x(t)$ are given by solving the system of anisotropic geometric evolution equations:

$$\partial_t x - \operatorname{div}_{\mathcal{M}}(A\,\nabla_{\mathcal{M}} x) = 0 \qquad (1)$$

We start with the initial condition $\mathcal{M}(0) = \mathcal{M}_0$. We define the tensor A such that we have strong diffusion along the surface features and weak diffusion across them. As before, we use our moment-based classifier $\mathcal{C}_\epsilon$ to detect features. When computing $\mathcal{C}_\epsilon$, we also obtain a basis w^1, w^2 in the tangent plane $\mathcal{T}_x\mathcal{M}$, defined by the major and medium eigenvectors of the first order moment. In this basis, the tensor A is defined as

$$A = \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{C}_\epsilon \end{pmatrix}$$

Given the above, the application of A to a vector x is:

$$A\,x := \left((x \cdot w^1)w^1 + \mathcal{C}_\epsilon(x \cdot w^2)w^2\right).$$

Since $\mathcal{C}_\epsilon$ is high in smooth regions and low close to edges and corners, Equation (1) smooths the surface by keeping the features. Due to the anisotropy A, we enforce a signal enhancement in the direction of the eigenvector w^1. In the direction of w^2, the diffusion is proportional with the classifier $\mathcal{C}_\epsilon$, i.e., strong in smooth areas and weak close to edges, which is exactly what we desire.

A problem occurs due to the diffusion in the tangential direction. Namely, points tend to drift on the surface, which may cause nonuniform point densities ultimately leading to holes in the surface. To prevent this, we project the velocity $\mathrm{div}_{\mathcal{M}}(A\,\nabla_{\mathcal{M}} x)$ onto normal direction.

Since the point set changes, we recompute the point normals after every few (1..3) smoothing steps. This implies recomputing the points' neighborhoods, since these may change due to the point displacements. As we already do the relatively expensive neighborhood computations, we also recompute the classifier and assemble the global stiffness matrix at this time.

Figure 1 shows several results, all obtained with a few tens of diffusion iterations. The important surface edges, such as the bunny's ear edges, body-hip contact line, the transversal cut of the femur, and the carved letters on the stone, are preserved. Small 'noise' details, such as the bunny's skin ripples, bone irregularities, and stone graininess, are removed.

One deformation step takes about one second on a point cloud of 65000 points on a Pentium IV PC at 1.8 GHz. In comparison, applying the same technique on triangle meshes, which uses a very similar implementation, we could process approximately the double number of points per second [8]. The reason for this difference in speed is that no neighborhood recomputing is needed in the mesh case, as the topology stays fixed. However, we noticed the point based fairing to be visibly more robust to large diffusion steps than its mesh based counterpart.

References

[1] A. Adamson and M. Alexa. Approximating and intersecting surfaces from points. In *Proc. Eurographics Symposium on Geometry Processing*, pages 89–97, 2003.

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

[3] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 39–48. ACM Press, 1998.

[4] J. D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984.

[5] J.-D. Boissonnat and J. Flototto. A local coordinate system on a surface. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 116–126. ACM Press, 2002.

[6] U. Clarenz, U. Diewald, and M. Rumpf. Nonlinear anisotropic diffusion in surface processing. *Proc. Visualization 2000*, pages 397–405, 2000.

[7] U. Clarenz, M. Rumpf, and A. Telea. PDEs on point clouds. *Bonn University, SFB 611, preprint*, 2004.

[8] U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. *to appear in Transactions on Visualization and Computer Graphics*, 2004.

[9] M. Desbrun, M. Meyer, P. Schroeder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, pages 317–324, 1999.

[10] M. Gopi, S. Krishnan, and C. T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Proc. Eurographics 2000*, volume 19(3), 2000.

[11] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 71–78, 1992.

[12] L. Kobbelt. Discrete fairing. In *Proceedings of the 7th IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1997.

[13] L. Linsen and H. Prautzsch. Global versus local triangulations. In *Proc. Eurographics 2001 (short presentations)*, pages 71–78, Manchestrer, UK, 2001.

[14] M. Pauly. *Point primitives for interactive modeling and processing of 3D geometry.* Dissertation, Department of Computer Science, ETH Zürich, 2003.

[15] G. Taubin. A signal processing approach to fair surface design. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 351–358, 1995.

[16] H. Xie, J. Wang, J. Hua, H. Qin, and A. Kaufman. Piecewise C^1 Continuous Surface Reconstruction of Noisy Point Clouds via Local Implicit Quadric Regression. In *Proc. IEEE Visualization 2003*, pages 198–206. ACM Press, 2003.

[17] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Computer Graphics (Proc. ACM SIGGRAPH 2001)*, pages 267–275, 2001.