Freeform Shape Representations for Efficient Geometry Processing

Leif Kobbelt Mario Botsch

Computer Graphics Group
RWTH Aachen

Abstract

The most important concepts for the handling and storage of freeform shapes in geometry processing applications are parametric representations and volumetric representations. Both have their specific advantages and drawbacks. While the algebraic complexity of volumetric representations $\mathcal{S} = \{(x, y, z) \mid f(x, y, z) = 0\}$ is independent from the shape complexity, the domain Ω of a parametric representation $\mathbf{f} : \Omega \rightarrow \mathcal{S}$ usually has to have the same structure as the surface $\mathcal{S}$ itself (which sometimes makes it necessary to update the domain when the surface is modified). On the other hand, the topology of a parametrically defined surface can be controlled explicitly while in a volumetric representation, the surface topology can change accidentally during deformation. A volumetric representation reduces distance queries or inside/outside tests to mere function evaluations but the geodesic neighborhood relation between surface points is difficult to resolve. As a consequence, it seems promising to combine parametric and volumetric representations to effectively exploit both advantages.

In this talk, a number of applications is presented and discussed where such a combination leads to efficient and numerically stable algorithms for the solution of various geometry processing tasks. These applications include surface remeshing, mesh fairing, global error control for mesh decimation and smoothing, and topology control for level-set surfaces.

1. Introduction

The field of geometry processing deals with the generation of and the efficient operation on all kinds of geometry representations. The design of efficient algorithms always has to be accompanied by the design of suitable data representations. Since the data to be processed are mainly geometric objects, each specific problem requires the "right" shape representation to be chosen in order to enable efficient access to the relevant information. In this context there are two major classes of surface representations: *parametric* surfaces and *volumetric* surfaces.

A parametric surface is an *explicit* representation, usually given by a function $\mathbf{f} : \Omega \subset \mathbb{R}^2 \rightarrow \mathcal{S} \subset \mathbb{R}^3$ that maps a two–dimensional parameter domain Ω into 3-space. In contrast, a volumetric surface is *implicitly* defined to be the zero–set of a scalar function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, i.e. $\mathcal{S} = \{(x, y, z) \in \mathbb{R}^3 \mid f(x, y, z) = 0\}$. From an abstract point of view, the first one can be considered the *range* of a function, while the latter one is defined to be the *kernel* of a function. Both representations have their own strengths and weaknesses, as we will see in the next sections.

1.1. Explicit Surfaces

An explicit parameterization $\mathbf{f}$ enables the reduction of several three–dimensional problems on the surface $\mathcal{S}$ to two–dimensional problems in the parameter domain Ω. For instance, points on the surface can easily be generated by simple function evaluations of $\mathbf{f}(u, v)$. Neighborhood queries on the surface $\mathcal{S}$ reduce to finding neighboring points in the parameter domain Ω. If the metrics in Ω and $\mathcal{S}$ are sufficiently similar we can therefore generate triangulations of $\mathcal{S}$ by lifting a 2D Delaunay triangulation [31] of Ω into 3–space.

Generating such parameterizations, on the other hand, can be quite hard and is an active area of research [9, 1, 15, 27], since the parameter domain Ω has to match the topological and metric structure of the surface $\mathcal{S}$. When changing the shape $\mathcal{S}$ it may even be necessary to update the parameterization accordingly in order to reflect the respective changes of the underlying geometry: a low–distortion parameterization $\mathbf{f}$ requires the metrics in $\mathcal{S}$ and Ω to be similar and hence we have to avoid or adapt to excessive stretching.

However, since the surface $\mathcal{S}$ is the range of the parameterization $\mathbf{f}$, the topology of the surface can explicitly be controlled. In contrast, there is no easy way to control the geometry in the embedded space, like e.g., preventing the surface from self–intersecting.

Usual data representations for explicit surfaces are piecewise linear polygonal meshes [7, 6] or higher order NURBS surfaces [30]. In both cases the storage complexity directly correlates with the geometric complexity of the surface $\mathcal{S}$ itself.

1.2. Implicit Surfaces

Since an implicit surface is defined to be the kernel of a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, geometric inside/outside queries for closed surfaces simplify to function evaluations of f and checking the sign of the resulting value. The function f is not uniquely determined, but the most common and most natural implicit representation is the so called *signed distance function* which maps each point $(x, y, z) \in \mathbb{R}^3$ to its signed distance from the surface $\mathcal{S}$. Negative function values designate points inside the object and positive values points outside the surface. This representation simplifies distance computations to simple function evaluations.

Implicit surfaces do not provide any means of parameterization. Hence, it is for instance almost impossible to consistently paste textures onto evolving implicit surfaces. However, they allow for the design of algorithms that are free of parameterization artifacts, since they are only based on intrinsic geometric properties of the surface.

One of the strengths of implicit surfaces is that they can easily change their topology, an important property in the context of constructive solid geometry. Unfortunately, this is also one of the main problems since there is no mechanism to prevent the topology from changing accidentally, i.e. to prevent the surface from merging or splitting, respectively. On the other hand since an implicit surface is a level–set of a potential function, geometric self–intersections cannot occur.

Because the parameter domain of the implicit function is the whole 3–space, the function f is usually restricted to some bounding box around the surface. The most basic representation is a uniform scalar 3D grid f_{ijk} of sampled values of f [26]. Function values in the interior of voxels are obtained by tri–linear interpolation. For this basic data structure the memory consumption grows cubically if we increase the precision by reducing the edge length of grid voxels.

Since the signed distance values are most important in the vicinity of the surface we can use a higher sampling rate only in these regions in order to achieve a more memory efficient data structure. Hence, instead of a uniform 3D grid an adaptive octree can be used to hold the sampled values [32]. Compared to the uniform grid this lowers the storage complexity from cubic to quadratic.

If we additionally restrict the local refinement to those cells where the tri–linear interpolant deviates more than a prescribed tolerance from the actual distance field, the resulting approximation adapts to the locality of the surface as well as to its shape [13]. Since extreme refinement is only necessary in regions of high surface curvature, this approach reduces the storage complexity even further and results in a memory consumption comparable to explicit representations.

Restricting the surface deformation to the direction of the local surface normal leads to the so called *level set surfaces* [34, 29]. Since the direction of motion is fixed, the surface evolution is fully determined by providing a scalar speed function, defining the amount of movement for each surface point at a certain time. Level set surfaces can be efficiently implementated using narrow–band or fast–marching techniques [33].

2. Conversion

As we have discussed in the last section, we should choose the suitable shape representation depending on the given problem if we are aiming at efficient algorithms. This requires conversion methods between explicit and implicit surface representations.

Since both kinds of representations are usually finite samplings, like e.g. polygonal meshes in the explicit case and uniform/adaptive grids in the implicit case, each conversion corresponds to a resampling step. Therefore we have to minimize the loss of information for these conversion routines.

An explicit surface can be constructed from a volumetric representation by extracting the zero–level iso–surface of the corresponding implicit function. The standard method for this contouring task is the Marching Cubes (MC) algorithm [26]. Since applying the MC algorithm corresponds to a regular resamping of f, high–frequency geometric details like, e.g., sharp edges or corners, will not be captured.

A refinement of the underlying 3D grid is not capable of solving this problem: in the vicinity of sharp features the surface is not differentiable, therefore the approximation power breaks down from quadratic to linear, resulting in a point–wise convergent surface with potentially diverging normals [22, 4]. However, the additional use of gradient information enables the robust detection and reconstruction of sharp features [22, 18], thereby minimizing the loss of information.

The conversion from explicit to implicit representation amounts to the generation of the signed distance function [20, 36, 38]. This can be done by voxelization, by fast–marching methods [33], or even by using graphics hardware [19]. To avoid alias–artifacts for a later back–conversion the respective gradient information should also be stored.

3. Applications

In this section we present a set of applications that exploit the strengths of both surface representations by choosing the most suitable one depending on the problem, leading to more efficient and even more robust geometry processing algorithms. We will also show that even better results can be achieved if we do not restrict to conversions between the different surface representations, but instead combine and use them simultaneously.

3.1. Remeshing

Polygonal meshes can be considered as piecewise linear parameterizations that represent each point in the interior of a triangle by a barycentric combination of its vertices. This kind of parameterization is competely defined by the distribution of vertices on $\mathcal{S}$ and their connectivity. We can rate the numerical quality of such a parameterization $\mathbf{f}$ by the shape of the respective triangles on the surface $\mathcal{S}$.

When using triangle meshes in numerical simulations, we have to guarantee that no degenerate faces are present since they have, e.g., no well defined normal vectors. Unfortunately, many tesselation algorithms in today's CAD systems generate this kind of meshes with sometimes extremely bad triangle quality. Therefore we have to find a high–quality tesselation, i.e. a new parameterization $\mathbf{f}^*$, for a geometry $\mathcal{S}$ given by a low quality parameterization $\mathbf{f}$, a task referred to as remeshing [9, 1, 15, 27].

Trying to perform the remeshing on the surface directly can be quite hard and special care has to be taken to end up with a numerically robust algorithm capable of handling all kinds of geometric as well as topological degeneracies [5].

Since we are only interested in the geometry of the given surface, we avoid all difficulties that emerge from the bad parameterization $\mathbf{f}$ by converting the mesh to an implicit representation. Since this only requires distance computations, it is not affected by degenerate triangles. The resulting implicit representation only contains relevant geometric information.

A new parameterization can now be generated by a second conversion step back from the implicit to an explicit representation, e.g. by the methods proposed in [22, 18]. The resulting polygonal mesh is an approximation to the original mesh and in addition provides a tesselation of higher quality. The approximation error can easily be bounded by choosing a sufficient grid resolution for the implicit representation.

This double conversion results in a fully automatic, numerically robust remeshing algorithm that is feature preserving and has bounded approximation error. The resulting meshes are guaranteed to be proper 2–manifold surfaces.

3.2. Mesh Fairing

Real world datasets acquired by, e.g., range scanning, medical imaging methods, or other physical measurements ususally contain a certain amount of measurement noise. Removing those high–frequency errors is usually referred to as mesh smoothing or mesh fairing.

No matter whether we follow the signal–processing approach [35], or an energy minimization method [21], or a PDE based flow technique [10], the standard fairing algorithm ends up to be some variant of Laplacian smoothing. This operator acts on the surface by taking a vertex and its one–ring neighbors into account. Unfortunately, the rate of convergence locally depends on the (ratio of) local edge lengths, leading to a significantly slower surface evolution in regions of higher sampling density.

Unless proper boundary constraints are imposed, there is always a trivial solution to the resulting system of equations, i.e. the mesh will eventually collapse to a single point. Even after a small time step one can observe a noticable shrinkage of the surface. Therefore different techniques have been proposed to prevent shrinkage by the use of (approximate) volume preservation [35, 10, 25].

If we use an implicit representation instead, we can define a level set smoothing operator that is purely based on the surface geometry [33]. The level set approach will provide us with curvature dependent or constant speed functions for the surface evolution, corresponding to mean curvature flow or dilation/erosion of the surface. Since all geometric entities are computed on a regular grid this approach is free of parameterization artifacts. In addition, exact volume preservation can be achieved by a correction term added to the speed function.

As in the last example, the final explicit surface is generated by a suitable contouring algorithm. Since the resulting surface is supposed to be smooth, there is no need to use a feature sensitive method, hence plain MC will be sufficient in this case.

3.3. Global Error Control

Most engineering applications require the processed surface to stay within a prescribed error tolerance. In the context of mesh decimation several highly efficient *local* error measures have been proposed [14, 24]. In contrast, computing the exact *global* error between two surfaces or constructing exact tolerance volumes/envelopes, respectively, is computationally very expensive, because the costs depend on the complexity of the surface shape [23, 8, 16].

An implicit representation based on the signed distance function provides the respective distance computations for free since they are just function evaluations, i.e. the complexity just depends on the approximation tolerance and

no longer on the shape complexity. Therefore it seems to be promising to combine an *explicit* mesh processing algorithm with an *implicit* global error representation. This has been done by Zelinka and Garland [39] using a uniform grid and a piecewise constant approximation of the implicit distance function.

Better approximation power and less memory consumption can be achieved with an adaptive grid and tri–linear approximation [13]. Switching from this piecewise tri–linear C^0 representation to a piecewise linear C^{-1} representation preserves the same asymptotic approximation properties, but the additional degrees of freedom lead to a better approximation with lower memory consumption [37].

The combination of an explicit decimation scheme with this implicit representation of the signed distance field provides high quality results satisfying a *global* approximation error at computational costs comparable to methods using *local* error estimates.

3.4. Level Set Surfaces with Topology Control

Although level set surfaces are already known for some time, they have been getting increasingly popular in the last few years [34, 29, 28]. In medical imaging applications they are used for segmentation and reconstruction of organic structures. In this context a certain a–priori knowledge about the structure, i.e. the topology, of the surface to be generated may exist, e.g. when reconstructing a cortical surface from MRT scans. But also in other applications we may have to prescribe the topology of the implicit surface or we at least want to prevent the topology from changing during the surface evolution.

Because of the lacking parameterization, implicit representations provide no means of topology control. Han et al. [17] overcome this problem by modifying the update rules for the level set computation in order to prevent the topology from changing.

We propose to combine the topology preserving properties of parametric surfaces with the parameterization–free implicit surface representation [2, 3]. The topology control is achieved by placing samples on the edges of the voxel grid whenever the surface is about to change its topology, e.g. when the surface is touching itself. Since these so called *cuts* are placed at the exact location of the critical points on the respective edges, we achieve sub–voxel accuracy for the boundary interface reconstruction.

These cuts are actually surface samples and hence can be considered as explicit surface information. This integration of explicit information into the implicit level set computation guarantees topology preserving surface evolutions, while keeping the algorithmic structure almost the same.

In a similar sense Enright et al. [12, 11] used particles as some kind of explicit representation in order to preserve surface characteristics and alleviate loss of mass during the deformation.

4. Conclusion

Both explicit and implicit surface representations have their own advantages and drawbacks. We have shown that exploiting this fact by choosing the "better" suited of these two representations depending on a specific problem leads to more robust and more efficient geometry processing algorithms.

Since each conversion between parametric and volumetric representations corresponds to a resampling of the respective explicit or implicit function, care has to be taken not to lose relevant high–frequency shape information.

Even better results can be achieved by not just converting between but also combining the two different representations and using them simultaneously.

We outlined a set of applications based on such a combination of implicit and explicit surface information, showing geometric algorithms that are free of parameterization artifacts and that have enhanced computational efficieny for global error bounds. This approach also provides a topology preserving mechanism for level set surfaces.

References

[1] P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. In *Siggraph 02 Conference Proceedings*, pages 347–354, 2002.

[2] S. Bischoff and L. Kobbelt. Parameterization free active countour models with topology control. In *4th Israel–Korean Binational Conference on Geometric Modeling and Computer Graphics*, pages 69–74, 2003.

[3] S. Bischoff and L. Kobbelt. Sub–voxel topology control for level–set surfaces. Submitted.

[4] M. Botsch and L. Kobbelt. Resampling feature and blend regions in polygonal meshes for surface anti-aliasing. In *Eurographics 01 Conference Proceedings*, pages 402–410, 2001.

[5] M. Botsch and L. Kobbelt. A robust procedure to eliminate degenerate faces from triangle meshes. In *Vision, Modeling, Visualization 2001 Proceedings*, pages 283–289, 2001.

[6] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. Openmesh – a generic and effi cient polygon mesh data structure. In *OpenSG Symposium*, 2002.

[7] S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed edges - a scalable representation for triangle meshes. In *ACM Journal of Graphics Tools 3 (4)*, 1998.

[8] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. P. Brooks, Jr., and W. Wright. Simplifi cation envelopes. In *SIGGRAPH 96 Conference Proceedings*, pages 119–128, 1996.

[9] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. In *Eurographics 02 Conference Proceedings*, pages 209–218, 2002.

[10] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature fbw. In *SIGGRAPH 99 Conference Proceedings*, pages 317–324, 1999.

[11] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, to appear.

[12] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *Siggraph 02 Conference Proceedings*, pages 736–744, 2002.

[13] S. Frisken, R. Perry, A. Rockwood, and T. Jones. Adaptively sampled distance fi elds: A general representation of shape for computer graphics. In *Siggraph 00 Conference Proceedings*, pages 249–254, 2000.

[14] M. Garland and P. Heckbert. Surface simplifi cation using quadric error metrics. In *SIGGRAPH 97 Conference Proceedings*, pages 209–216, 1997.

[15] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. In *Siggraph 02 Conference Proceedings*, pages 355–361, 2002.

[16] A. Gueziec. Surface simplifi cation inside a tolerance volume. In *SecondAnnual International Symposium on Medical Robotics and Computer Aided Surgery, November 1995*, pages 132–139, 1995.

[17] X. Han, C. Xu, and J. Prince. A topology preserving deformable model using level sets. In *Computer Vision and Pattern Recognition Proceedings 01*, pages 765–770, 2001.

[18] T. Ju, F. Lasasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Siggraph 02 Conference Proceedings*, pages 339–346, 2002.

[19] I. K. E. Hoff, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Siggraph 99 Conference Proceedings*, pages 277–286, 1999.

[20] A. Kaufman. Effi cient algorithms for 3d scanconversion of parametric curves, surfaces, and volumes. In *Siggraph 87 Conference Proceedings*, pages 171–179, 1987.

[21] L. Kobbelt. Discrete fairing. In *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces*, pages 101–131, 1996.

[22] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *Siggraph 01 Conference Proceedings*, pages 57–66, 2001.

[23] L. Kobbelt, S. Campagna, and H.-P. Seidel. A general framework for mesh decimation. In *Graphics Interface*, pages 43–50, 1998.

[24] P. Lindstrom and G. Turk. Fast and memory effi cient polygonal simplifi cation. In *IEEE Visualization Conference Proceedings*, pages 279–286, 1998.

[25] X. Liu, H. Bao, H.-Y. Shum, and Q. Peng. A novel volume constrained smoothing method for meshes. In *Graphical Models*, 2002.

[26] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In *Siggraph 87 Conference Proceedings*, pages 163–170, 1987.

[27] B. Lvy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Siggraph 02 Conference Proceedings*, pages 362–371, 2002.

[28] K. Museth, D. E. Breen, R. T. Whitaker, and A. H. Barr. Level set surface editing operators. In *SIGGRAPH 2002 Conference Proceedings*, pages 330–338, 2002.

[29] J. S. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[30] L. A. Piegl and W.Tiller. *The NURBS Book*. Springer, 1997.

[31] F. P. Preparata and M. I. Shamos. *Computational geometry: an introduction*. Springer, 1985.

[32] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison–Wesley, 1994.

[33] J. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Science*, volume 93, pages 1591–1595, 1996.

[34] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[35] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 95 Conference Proceedings*, pages 351–358, 1995.

[36] L. Velho and J. Gomez. Approximate conversion of parametric to implicit surfaces. In *Eurographics 96 Conference Proceedings*, pages 327–337, 1996.

[37] J. Wu and L. Kobbelt. Piecewise linear approximation of signed distance fi elds. Submitted.

[38] G. Yngve and G. Turk. Robust creation of implicit surfaces from polygonal meshes. In *IEEE Transactions on Visualization and Computer Graphics*, pages 346–359, 2002.

[39] S. Zelinka and M. Garland. Permission grids: Practical, error-bounded simplifi cation. In *ACM Transactions on Graphics*, pages 207–229, 2002.