Blending Operations Using Rolling-Ball Filleting

Ibraheem Alhashim
School of Computing Science
Simon Fraser University

April 14, 2009

Abstract

Blending sharp edges in solid models is an important task of Computer-aided design. The prominent algorithms for producing blending fillet surfaces relay mostly on the rolling-ball method. These algorithms are applied on B-Rep solid models that are used in many CAD/CAM systems. Such solid models allow for a hierarchical approach of modeling which can incorporate blending operations within its structure. Blending on arbitrary meshes, however, falls into a different domain and is a challenging problem. This report describes an implementation of an edge blending algorithm applied to regular polygonal surfaces.

1 Introduction

Blending and rounding sharp edges is an important aspect in CAD. Such operations provide a smooth and continuous transition from one surface to another. The process is important in many situations including manufacturing and engineering [7]. A number of manufacturing processes, such as casting, require blended edges. Smooth edges also address safety concerns for sharp corners. Round edges relate to engineering performance problems such as aerodynamic drag and low-radar cross section issues [2]. Most algorithms used in commercial CAD/CAM systems that generate blends are not open source and are not well known. Since the earliest proposals by Choi, et. al [4], research for finding algorithms that can deal with all the different cases of blending remains an on going one.

2 Background

Most solutions derived for the blending problem are solutions that relate to free-form parametric surfaces and B-rep solid models. This section describes the widely used rolling-ball method and presents the most recent work relating to blending operations on B-rep.

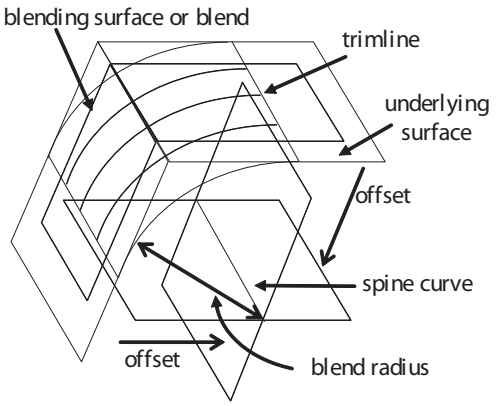

Figure 1: Terminology of blending [5]

2.1 Rolling-Ball Blending

The rolling-ball method appears to be the common method for creating blends between two surfaces. The general algorithm involve operations of surface intersection, section curve construction and curve sweeping [1][3]. An imaginary ball of some radius is rolled along the intersection between two surfaces in such a way that it remains in simultaneous contact with both surfaces at points of tangency. The arcs on the rolling ball between the two contact curves are called the section curves of the blending surface. When these section curves sweep along the center curve, also called the spine curve, they form the blending surface. The shared boundary between the underlying surfaces and the blend surface is called the trim-line. Figure 1 displays the basic terminology used in the rolling-ball blending algorithm.

Although being the most widely used method, rolling-ball suffers from several problems. One of the problems is that it produces self-intersecting blends between surfaces of small radii [3]. Another drawback is producing surfaces with curvatures that do not agree with the adjoining surfaces [2]. Also, applying roll-blending alone does not generate the desired edge rounding for the entire model. Other cases in which problems appear are on corners, terminating edge blend problems, and others relating to global blending. These cases and several others have been discussed in more detail by Matsuki, et al. [5].

2.2 Extended Regular Blending

To help overcome some of the main challenges of blending, Matsuki, et al. proposed extending the rules of B-rep solid models to include surface modeling techniques. Their algorithm involves defining enlarged-surfaces sequentially in order to create what they called extended regular blending. Blending surfaces around sharp edges can be generated by following three steps. First, two sets of continuously connected surfaces are removed from the object. Secondly, blend-

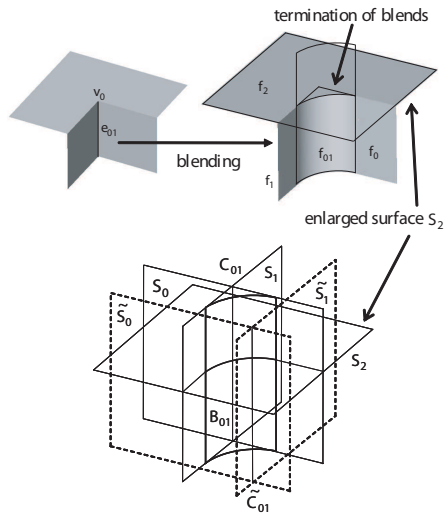

Figure 2: Construction of terminating edge for an extended regular blending

ing surfaces are then defined by simply repeating regular blending, generated by rolling-ball, of each surface pairs and the valid parts of the blending surfaces are then connected. Finally, all valid pieces are merged into the original solid. Figure 2 shows one general blending problem addressed by extended regular blending. Matsuki, et al. work is one of the most recent research work involving blending operations with B-rep.

3 Blending Polygonal Surfaces

Constructive solid geometry (CSG) and B-rep are the most commonly used representations in CAD/CAM systems. Blending operations in these system require exact details about the edges of the model. However, these operations are not trivial when used on polygonal surfaces. Figure 3 shows a model in two different representations using two different softwares that are intended for different users.

Research by Wang, et al. have explored one approach that apply to irregular meshes of an arbitrary topological type using Varudy patches[8]. The idea of their approach treats blending as a surface fitting problem that can be solved by first restructuring a mesh to obtain a new mesh suitable for curves construction, and then filling all the holes with n-sided Varady patches. They state that their algorithm can help in blending, smoothing and interpolation problems on polygonal meshes. Another algorithm that has the potential of performing blends on irregular meshes may involve subdivision and the laplacian surface editing technique presented by Sorkine, et al [6].

In this section we describe our simple implementation of the rolling-ball

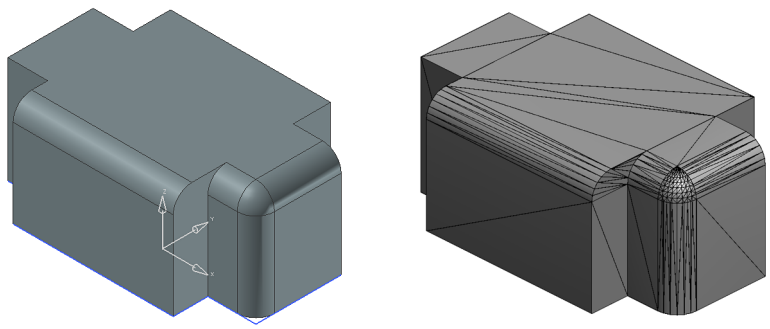

Figure 3: Different representations of the same model. *Left: using the Siemens UGS NX 6.0 CAD/CAM system. Right: using the Autodesk 3ds Max modeling tool.*

filleting method for triangular meshes.

3.1 Concept

The idea behind our implementation is to try and construct blending surfaces around sharp corners. Such corners in B-rep are easily identified, however, in triangular mesh they are not trivially apparent and one may relay on edge detection algorithms to find them. We assume that the edges are predefined by a user at the start of the blending process. The resulting blending surface can be either derived from the mesh structure by refining the area around the edge or it can be constructed and then stitched onto the surface of the mesh. The latter method is assumed in this implementation.

3.2 Blending Algorithm

Given a set of vertices along the sharp edge, the basic elements used by the rolling-ball algorithm needs to be computed. The first element is the trimlines on both virtual surfaces. These lines are represented as points on the surface that would serve as the cutting lines on the mesh and the starting point of the blending surface. The density of the points depends on the number of vertices in the preselected vertex list. One can increase the density of the points to achieve smoother results using subdivision on the expense of creating many extraordinary points along the trimlines. One possible approach to resolve these problems is by using modified subdivision algorithms such as the one described by Zorin [9] after the blending operation is over.

After identifying the trimlines, the next step would be to construct the spine curve needed for the rolling-ball method. The curve is approximated to a number of points that lay on the plane defined by the two trimline points and the selected vertex on the sharp edge. Next, our algorithm constructs the blending surface by creating triangular segments between each two trimline points.

These segments represent the blending surface and are constructed by following a vector, centered at the spine curve, starting from one trimline to the other. The construction step can be modified to account for constant and variable-sized blends by moving the trimline points along the virtual surfaces.

Of course the important last step is to stitch the blending surface with the original mesh. However, this step is not implement due to time constrains.

3.3 Problems and Limitations

After working on blending surfaces on triangular meshes, it is clear that the same problems found in B-rep blending operations apply to triangular meshes. More difficulties appear in triangular meshes since the boundary of the two virtual surfaces is not defined and is subject to abrupt changes. Also, the algorithm does not perform well on meshes with bad triangulations, therefore, a prepossessing of the mesh is needed in order to achieve better results.

Edge terminations and corner cases are complicated to deal with in triangular meshes. Adding caps at the end of the blending surface may result in substantial change to the original model. Other problems relating to the nature of triangular meshes is the fact that each applied blending operation changes the connectivity of the mesh making it more complex to add other blends on other parts of the mesh. Therefore, blending different parts of the mesh needs to be done while keeping in mind the global changes to the mesh's connectivity. A possible solution for this is to re-triangulate and refine the mesh in a way that helps with the required successive blending operations.

4 Results

The algorithm seems to produce reasonably good results for regular triangular meshes. Such an algorithm can be incorporated into a general purpose modeling software as an object modifier. Most modeling systems use subdivision to help blend sharp edges, however, these operations involve the entire mesh structure and are therefore inefficient for small local blending operations.

Our implementation assumes regular triangulation along the trimlines. In such cases the produced blending surface is almost equivalent to the ones generated with B-rep representation. Figure 4 shows a blending operation performed on one of the sharp edges of a triangular mesh. Dealing with edge termination and corner cases are not yet considers and further research is thus needed.

5 Possible Usage

Blending in triangular meshes is primarally used for aesthetic purposes. Another possible use of blending algorithms on triangular meshes is in interactive real-time environments such as video games. Game engines are limited by the average capabilities of GPUs available in the market. Thus, the focus of rendering detailed objects in the scene is usually shifted towards the active parts of the

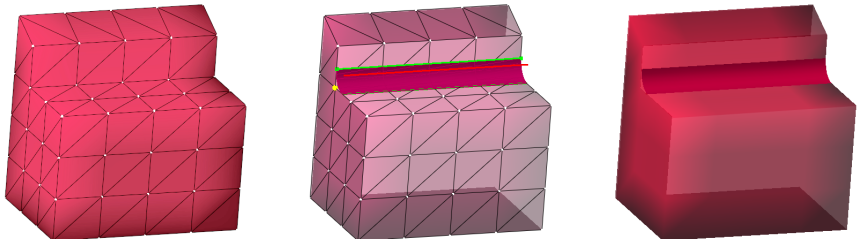

Figure 4: Results on a triangular mesh using our blending algorithm.

environment such as characters and game items. This leaves other parts such as walls, roofs, and other decorative objects as being modeled with minimum triangle counts to achieve higher rendering speeds.

One can create a special data structure for boxes or walls that can dynamically change its shape around edges to achieve better looking scenes. The idea is that the closer the object is, the more apparent that on its corners blending surface needs to be seen. Such approach can be thought of as a "LOD algorithm for walls". Figure 5 shows a simple sketch of how such structure would behave.

6 Conclusion

In this report we presented results of constructing blending surfaces using the rolling-ball algorithm that is applied to triangular meshes. We have presented the basic ideas used for generating blending surfaces on B-rep, briefly described the most recent work of using extended regular blending, and listed some of the problems relating to the blending operations. No solution exists yet for effectively generating all cases of blending, thus, more research is still needed to find better, more robust blending algorithms for B-rep and polygonal surfaces.

Acknowledgment

The implementation has been done using code from: the CGAL Open Source Project, the libQGLViewer OpenGL 3D viewer by Gilles Debunne, and different code snippets from the CGAL community.

References

[1] Jung-Hong Chuang and Wei-Chung Hwang. Variable-radius blending by constrained spine generation. *The Visual Computer*, 13(7):316–329, 1997.

[2] T. A. Grandine and T. A. Hogan. C2 pseudo rolling ball filleting. *Computing*, 79(2):163–173, 2007.

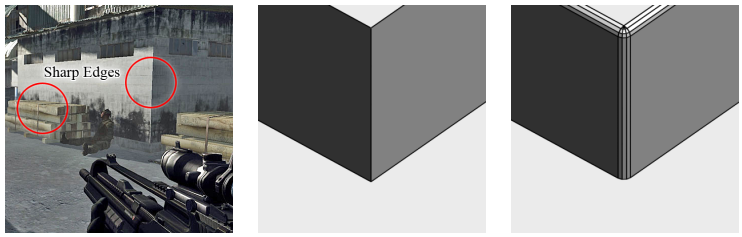

Figure 5: *Left: In game graphics CryEngine2 from Crytek (2007). Center: structure of the wall when looked at from far away. Right: a more detailed edge is generated when the viewer is closer to the wall.*

[3] Ling Huang and Xinxiong Zhu. Construction of blending surfaces. Technical Report HZ-TMSurf-Huang04, Beijing University of Aeronautics & Astronautics, 2000.

[4] Choi B. K. and Ju S.Y. Constant-radius blending in surface modeling. *Computer Aided Design*, 21(4):213–220, 1989.

[5] Norio Matsuki, Yoshiyuki Furukawa, and Fumihiko Kimura. An enlarged-surface based solid model for robust blending. pages 389–396, 2008.

[6] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM.

[7] Martin R. R. Vida, J. and T. Varady. A survey of blending methods that uses parametric surfaces. *Computer Aided Design*, 26:341–365, 1994.

[8] Xuefu Wang, Fuhua (Frank) Cheng, and Brian A. Barsky. Blending, smoothing and interpolation of irregular meshes using n-sided varady patches. In *SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 212–222, New York, NY, USA, 1999. ACM.

[9] Denis Zorin. Constructing curvature-continuous surfaces by blending. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 31–40, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.