CAD/Graphics Technology and Its Applications

— Proceedings of 8th International Conference on CAD/Graphics

Editors: Enhua Wu, Hanqiu Sun, Dongxu Qi

29-31 October 2003
Taipa, Macao, China

Sponsored by
China Computer Federation
National Natural Science Foundation of China
Macao Foundation

Organized by
Macao University of Science and Technology
University of Macau
The Chinese University of Hong Kong

Silhouette Smoothing by Boundary Curve Interpolation

Bin Wang[1,2], Wenping Wang[2], Johnson Wu[2], Jiaguang Sun[1]
1.Tsinghua University
2.The University of Hong Kong
E-mail: bwang@csis.hku.hk, wenping@csis.hku.hk
sowu@hkuspace.hku.hk, jgsun@tsinghua.edu.cn

Abstract

Conspicuous and visually unpleasant polygonal silhouettes are often produced when rendering a polyhedral approximation (i.e. a polygonal mesh) of a smooth surface model, regardless whatever shading method is used to improve the appearance of the interior region. We present in this paper a technique for displaying a coarse polygonal mesh with smooth silhouette. Unlike other existing methods that assume the availability of a fine model, our method uses only a coarse mesh to generate smooth silhouette through efficient boundary curve interpolation and local re-meshing near the reconstructed smooth silhouette. The time used by our method to render a polygonal mesh with smooth silhouette is about 80% *more than rendering the original mesh without any fix to the polygonal silhouette.*

Keywords: polygonal mesh, silhouette smoothing, Hermite interpolation, re-mesh

1 Introduction

It is a standard practice in interactive computer graphics to use polygonal meshes to represent curved surfaces for rendering. Standard shading methods include the Gouraud shading method and the Phong shading method, which are based on color interpolation or normal vector interpolation. These methods give a smooth illumination to the interior region of the polygonal mesh; the silhouette, however, still remains conspicuously polygonal, thus defeats the purpose of rendering the mesh as a smooth surface model. The challenging problem of generating a smooth silhouette is well recognized by the computer graphics community, as summarized by Foley et al. [3]: "No matter how good an approximation an interpolated shading model offers to the actual shading of a curved surface, the silhouette edge of the mesh is still clearly polygonal."

There are several ways of circumventing the difficulty in generating smooth silhouettes. In order to render a smooth curved surface model from its polyhedral mesh approximation, one may try to build a mesh with a large number of triangles so that the silhouette would look smooth enough. This approach is feasible with the development of faster graphics chips, but would mean that much extra computing resources are needed to generate and render more triangles in the interior of the mesh, a task that does not contribute to solving the silhouette smoothing problem. Moreover, generating a good quality refined mesh from a given coarse mesh is another nontrivial problem.

In this paper we present a new method for rendering a coarse polygonal mesh with smooth silhouettes. We assume that only the coarse mesh is available and no large fine mesh needs to be generated. We work with only the silhouette edges and triangles adjacent to the silhouette edges. The flow chart of the method is shown in Figure 1. There are six main steps: (1) We extract the visible silhouette edges and the associated triangles of the coarse mesh. (2) The silhouette edges are projected to the 2D viewing plane to give a series of line segments. (3) The straight line segments are replaced by a smooth boundary curve using Hermite interpolation. (4) We assume that these smooth interpolation boundary curves are the projection of the silhouettes of the smooth surface to which the coarse mesh approximates. Therefore we project these smooth boundary curves back to 3D space to give the smooth silhouettes of the mesh. (5) The smooth silhouette curves in 3D are sampled for local re-meshing. (6) The triangles resulting from the re-meshing, which are adjacent to the smooth silhouette edges, are rendered.

Because new vertices are added only on the silhouettes for re-meshing the model near the silhouette, there is only a small increase in the total number of the triangles to be rendered. Our experiments show that this method takes about 80% more time to generate and render a polygonal mesh with smooth silhouette than without smooth silhouette.

The remainder of this paper is organized as follows. In Section 2 we review related work. In Section 3 the new algorithm for silhouette smooth is described in detail. In Section 4 experimental results are presented. In Section 5 we conclude the paper with some remarks and problems for further research.

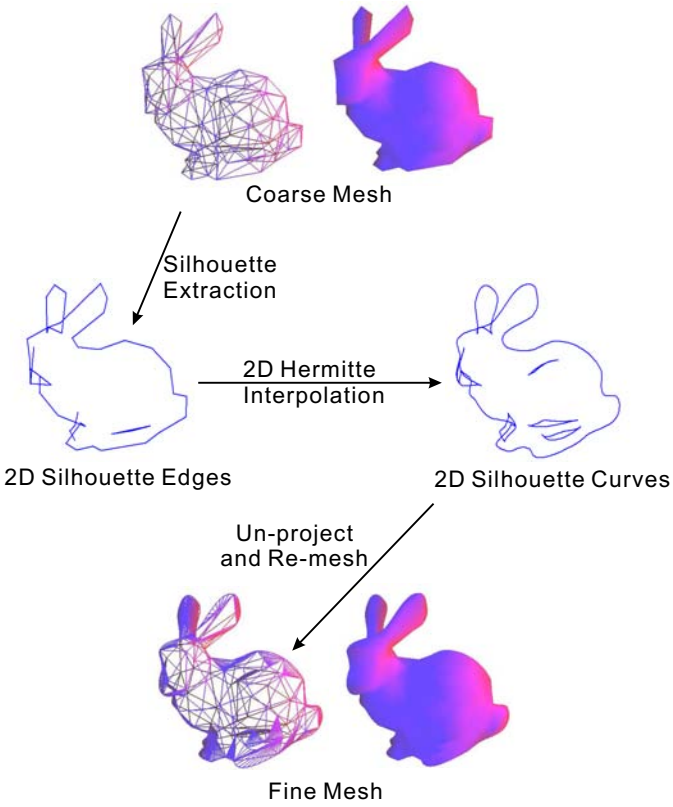


Figure 1. The flowchart of the silhouette smoothing procedure.

2 Related Work

There are mainly two approaches in the existing research aiming at producing a smooth silhouette of a polygonal mesh: the level of detail (LOD) techniques and silhouette clipping.

The level of detail technique (LOD) was first used for rendering terrain scenes. The basic strategy of LOD is to use a detailed mesh when the object is near the viewer, and to use a coarse mesh when the object is far away from the viewer [4]. Since the progressive mesh (PM) representation [6, 7] and some simplification algorithms [10, 14] were introduced for view-dependent LOD, the strategy of LOD has become more sophisticated, and has been widely extended [1, 11]. The PM represents a continuous spectrum of meshes; only the part of an object in the view frustum that is near the viewer and front-facing the viewer is displayed with fine details. Note, however, that the PM (or LOD) strategy is based on a complex representation of progressive meshes which requires a very fine original mesh to start with.

Silhouette clipping [13] assumes that both a coarse mesh and a fine mesh of a surface are available. It first renders a coarse mesh and then uses the stencil buffer to clip the rendering of the coarse mesh against the 2D smooth silhouette generated from the fine mesh. This technique produces good rendering result, and entails a complicated processing of the fine mesh; it needs to build a series of special progressive hulls: $v(M^0) \supseteq v(M^1) \ldots \supseteq v(M^n)$, where M^n is the original fine mesh, M^0 is the coarsest mesh, and $v(M^0)$ denotes the set of points interior to M. In contrast, we aim at producing a smooth silhouette from only a coarse mesh.

3 The Algorithm

In this section we present the details of our algorithm.

3.1 Silhouette Extraction

Silhouette is a useful feature for representing the shape of an object. Many researchers have studied fast silhouette extraction of polyhedral models. Markosian et al. [8] use a random algorithm to find silhouette. They first find some initial silhouette edges by testing a small subset of the edges and trace them to find more silhouette edges. Markosian et al's algorithm is fast, but it is not a deterministic algorithm for finding all the silhouette edges. Benichou and Elber [2] use the Gaussian sphere to find the silhouette of a polygonal mesh under a parallel projection. Hertzmann [5] and Pop et al. [12] solve this problem in dual space. Sander [13] finds the silhouette by locating an anchored cone of normal vectors in an n-ray tree. Some of these algorithms have an $O(\sqrt{n})$ complexity for an polyhedral mesh with n edges [13]. All these silhouette extraction algorithms are reasonably fast and can be used in our method. We use an exhuastive search in our method to find all silhouette edges. Experimental results show that the time of finding the silhouette edges is not the bottle-neck of the whole computation process.

3.2 2D Hermite Interpolation

Suppose that the silhouette edges of a mesh surface model have been extracted. Then these silhouette edges can be projected into line segments on the 2D viewing plane. To obtain a smooth silhouette boundary, we use a cubic Hermite curve to interpolate the two endpoints of each line segment. A Hermite interpolation curve needs two endpoints as well as the two tangent vectors at the two endpoints. The equation of a cubic Hermite interpolation is

$$X(u) = (2X_0 - 2X_1 + X_0' + X_1')u^3 - (3X_0 - 3X_1 + 2X_0' + X_1')u^2 + X_0'u + X_0,$$

$u \in [0, 1]$, where X_0 and X_1 are the two endpoints, and X_0' and X_1' are the tangent vectors at X_0 and X_1, respectively.

To define a cubic Hermite curve, we need to determine the directions and the magnitudes of the tangent vectors X_0' and X_1'. We project the normal vectors at

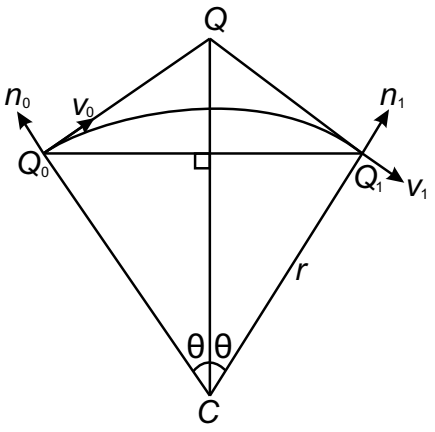


Figure 2. Approximate a Circular arc with a Hermite curve.

the vertices of the 3D mesh onto the $2D$ viewing plane, and rotate them by $\pi/2$ to get the directions of the end tangent vectors. Here the normal vector of a vertex of the mesh in 3D space is computed as a weighted average of the normal vectors of adjacent triangles sharing the vertex.

The magnitudes of the tangents are selected so that the resulting Hermite curve best approximates a circle if the end data points are from a circle, which is a problem considered in [9]. Figure 2 shows a situation where we want to approximate a circular arc with a Hermite curve. Let Q_0 and Q_1 be points in a circular arc of a circle with radius r and center C. Let θ be half of the angle subtended by the circular arc. Let $\mathbf{n}_0$ and $\mathbf{n}_1$ denote the normal vectors of this circular arc at Q_0 and Q_1, with corresponding tangent vectors $\mathbf{v}_0$ and $\mathbf{v}_1$. It can be shown [9] that, in order to best approximate the circular arc, the two tangent vectors should be set to be

$$\mathbf{v}_0 = 4\rho(Q - Q_0), \quad \mathbf{v}_1 = 4\rho(Q_1 - Q),$$

where $\rho = \cos\theta(1 + \cos\theta)$.

We use the same way as above to determine the magnitudes of the tangent vectors in the general case, i.e. when the end data points are not necessarily from a circle. With the tangent vectors completely determined above, we obtain a smooth Hermite curve. This curve is then projected back in the $3D$ space to give the smooth silhouette curve on the underlying smooth surface. Thus we get the smooth $3D$ silhouette from the coarse mesh.

3.3 Re-mesh and Rendering

Given a smooth $3D$ silhouette, re-meshing is required for rendering the coarse mesh. Let ABC be a front-facing triangle and BDC be a back-facing triangle. Let BC be a silhouette edge and BEC be the associated smooth silhouette curve. Let $P_0, P_1, \ldots P_n$ be sampled points on the silhouette curve. Our re-meshing scheme splits an original triangle into n triangles denoted by $\triangle AP_iP_{i+1}$ or $\triangle P_iP_{i+1}D$, where $i = 0, 1, \ldots, n-1$. Figure 4 shows the effect of applying this re-meshing scheme to a tetrahedron, which is expected to appear as a sphere due to silhouette smoothing.

For rendering the new mesh, the normal vector of the vertex P_i is calculated by the following linear interpolation

$$\mathbf{n}(t) = (1-t)\mathbf{n}_0 + t\mathbf{n}_1, \quad t \in [0, 1],$$

where $\mathbf{n}_0$ and $\mathbf{n}_1$ are respectively the normal vectors at the two endpoints of the original silhouette edge.

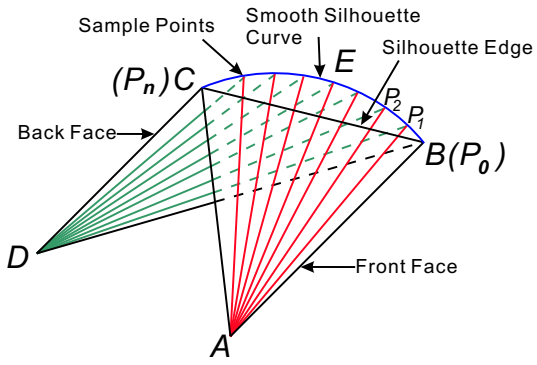
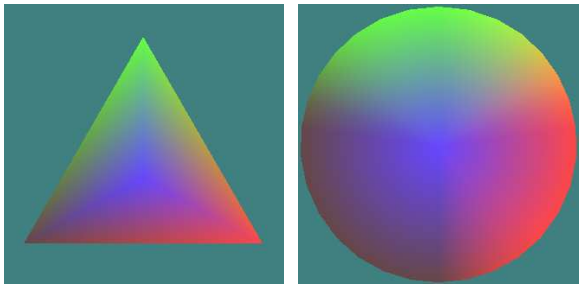


Figure 3. The re-meshing scheme.



(a) (b)

Figure 4. (a): A tetrahedron as a coarse mesh. (b): A smoothed mesh generated by our re-meshing scheme.

4 Results and Discussion

We present some test examples of using our silhouette smoothing algorithm. Figures 5(a) through 8(a) show different coarse meshes, which have 500 (Teddy), 1000 (cow), 200 (Venus), and 180 (sphere) triangles, respectively. Figures 5(b) and 5(c) show these meshes after silhouette smoothing, where 5(b) uses 3 sample points (including the end points) on every Hermite curve and 5(c) uses 5 sample points. For the purpose of comparison, Figure 5(d) shows a fine mesh containing 3192 triangles. It is evident that the silhouette of 5(c) is almost as smooth as that of 5(d).

Since the cow mesh has 1000 triangles, which is rather large for the small image size, the difference between Figures 6(a) and 6(b) is not obvious. However, with the close-up views in 6(d) for the smoothed silhouette and 6(c) for the non-smoothed one, the former looks clearly better than the latter.

Figure 7 shows a 200-triangle mesh (7(a)), a silhouette-smoothed mesh (7(b)), and a 5672-triangle fine mesh (7(c)). Although 7(b) has a smooth silhouette, its lack of interior details is also clear, if compared with 7(c); the same observation can be made by comparing Figures 5(c) and 5(d). That is because our silhouette smooth method only improves the smoothness of the silhouette without adding details to the interior region.

Figure 8 shows a coarse sphere mesh textured with a world map and a silhouette smoothed version of the same mesh with the same texture. Linear interpolation was used to compute the texture coordinates of the new vertices of the silhouette smoothed mesh.

Table 1 shows the timing data for rendering several models with our method. We observe that the time for rendering a mesh with smoothed silhouette is about 80% more than rendering the original coarse mesh, and in many cases this time is shorter than rendering a fine mesh that gives a comparable smooth silhouette. Therefore, our method can be considered an efficient method for improving silhouette smoothness of a coarse mesh when a more refined mesh is not available. All the timing data were taken on a PC with 2.4G Hz P4 CPU and a GF4-MX440 video card.

Table 1. Timing Data.

Teddy Bear	Time #1	Time #2	Time #3
Coarse mesh (500 triangles)	1.09	–	–
Silhouette smoothing (3 sample points)	1.72	0.15	0.38
Silhouette smoothing (5 sample points)	2.10	0.15	0.67
Fine mesh (3192 triangles)	4.15	–	–
Cow	**Time #1**	**Time #2**	**Time #3**
Coarse mesh (1000 triangles)	1.71	–	–
Silhouette smoothing (4 sample points)	3.31	0.30	1.14
Fine mesh (5804 triangles)	7.85	–	–
Venus	**Time #1**	**Time #2**	**Time #3**
Coarse mesh (200 triangles)	0.72	–	–
Silhouette smoothing (5 sample points)	1.14	0.08	0.23
Fine mesh (5672 triangles)	7.04	–	–
Sphere	**Time #1**	**Time #2**	**Time #3**
Coarse mesh (180 triangles)	0.66	–	–
Silhouette smoothing (5 sample points)	0.92	0.05	0.11

Time #1: Overall Time (ms/frame);
Time #2: Silhouette Computation Time (ms/frame);
Time #3: 2D Hermite Interpolation Time (ms/frame).

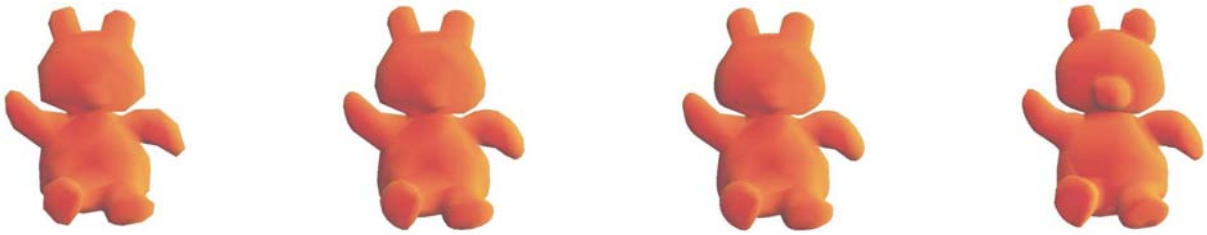
5 Concluding Remarks

We have presented a method for fast rendering of a polyhedral mesh with smooth silhouette. The method uses 2D Hermite curve interpolation to reconstruct a smooth silhouette edges of the mesh. Without assuming the availability of a fine mesh, our method produces satisfactory results, at moderate increase of processing time. In general, our algorithm converts polygonal silhouette edges into smooth curve silhouettes. When a 3D polyhedral model has sharp crease features that should not be converted to smooth boundaries, these features can be tagged in the data file so that they can be skipped by our algorithm.

The meshes used in this paper were drawn using the normal OpenGL format. However, it is found that, when meshes are drawn with the aid of vertex arrays, which is a more elaborate drawing mode commonly used for drawing large meshes, the time of our current implementation can be about 3 times more than the time of drawing only the coarse mesh without any fix to the polygonal silhouette. This provides the motivation for our current work on further speedup of our method.

A possible approach that may lead to a faster method is to obtain sample points on the smoothed silhouette without having to go through Hermite curve interpolation in the 2D viewing plane. This may be achieved by either a 3D Hermite curve interpolation scheme or a local operation of constructing an interpolatory subdivision surface to refine the mesh near the silhouette.

The coherence of the silhouette computed by our method cannot be ensured when the model is displayed with a moving view point. We anticipate that this problem can solved by performing a local interpolatory subdivision in a neighborhood of silhouette edges so that the smooth silhouette is computed in a view-independent manner. We plan to investigate this approach in our future research.



(a) (b) (c) (d)

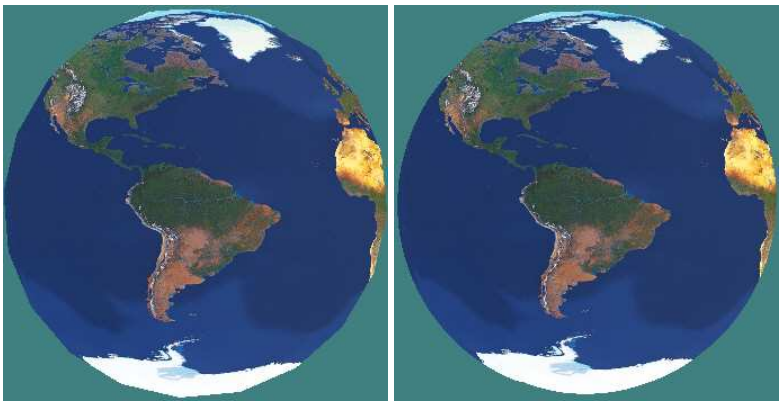
Figure 5. Teddy Bear mesh: (a) Coarse Teddy Bear mesh (500 triangles); (b) Mesh after silhouette smoothing (3 sampled points on 2D Hermite curve); (c) Mesh after silhouette smoothing (5 sampled points); (d) Fine Teddy Bear mesh (3192 triangles).

(a) (b) (c) (d)

Figure 6. Cow mesh: (a) Coarse Cow Mesh (1000 triangles); (b) Mesh after silhouette smoothing (4 sampled points); (c) An enlarged view of (a); (d) An enlarged view of (b).

(a) (b) (c)

Figure 7. Venus mesh: (a) Coarse Venus mesh (200 triangles); (b) Mesh after silhouette smoothing (5 sampled points); (c) Fine Venus mesh (5672 triangles).



(a) (b)

Figure 8. Sphere Mesh: (a) Coarse Sphere Mesh (180 triangles); (b) Mesh after silhouette smoothing (3 sampled points).

References

[1] D. I.Azuma, Daniel N. Wood, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle, View-dependent refinement of multiresolution meshes with subdivision connectivity, In *Proceedings of AFRIGRAPH 2003*, 2003.

[2] F. Benichou and G. Elber, Output sensitive extraction of silhouettes from polygonal geometry, In *Proceeding of Pacific Graphics 1999* , 60–69, October, 1999.

[3] J. D. Foley, A. van Dam, S. Feiner and J. Hughes, *Computer Graphics: Principles and Practice*, Second Edition, Addison-Wesley, 1990.

[4] P. Heckbert and M. Garland, Survey of polygonal surface simplification algorithms, In *SIGGRAPH 1997 Course notes* #25, August, 1997.

[5] A. Hertzmann and D. Zorin, Illustrating smooth surfaces, In *Proceedings of SIGGRAPH 2000* , 517–526, July, 2000.

[6] H. Hoppe, View-dependent refinement of progressive meshes, In *Proceedings of SIGGRAPH 1997* , 189–198, August, 1997.

[7] H. Hoppe, Smooth view-dependent level-of-detail control and its application to terrain rendering, In *Proceedings of Visualization 98, IEEE* , 35–42, October, 1998.

[8] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D.Bourdev, D. Goldstein and J. F. Hughes, Real-time non-photorealsitic rendering, In *Proceedings of SIGGRAPH 1997* , 415–420, August, 1997.

[9] M. E. Mortenson, *Geometric Modeling*, 2nd edition, Wiley and Sons, 1997

[10] D. P. Luebke and C. Erikson, View-dependent simplification of arbitrary polygonal environments, In *Proceedings of SIGGRAPH 1997*, 199–208, August, 1997.

[11] R. Pajarola, FastMesh: Efficient View-dependent Meshing, In *Proceedings of Pacific Graphics 2001*, 22–30, October, 2001.

[12] M. Pop, G. Barequet, C. A. Duncan and M. T. Goodrich, Efficient perspective-accurate silhouette computation and applications, In *Proceedings of the 17th annual symposium on Computational Geometry(SCG'01)*, 60–68, 2001.

[13] P. V. Sander, X. Gu, S. J. Gortler, H. Hoppe and J. Snyder, Silhouette Clipping, In *Proceedings of SIGGRAPH 2000*, 327–334, July, 2000.

[14] J. Xia and A. Varshney, Dynamic view-dependent simplification for polygonal models, In *Proceedings of Visualization 96, IEEE*, 327–334, October, 1996