Computing Subdivision Surface Intersection

S. Lanquetin, S. Foufou, H. Kheddouci, M. Neveu
LE2I, FRE CNRS 2309
UFR des Sciences et Techniques
Université de Bourgogne, BP 47870
21078 DIJON Cedex
sandrine.lanquetin@u-bourgogne.fr

ABSTRACT

Computing surface intersections is a fundamental problem in geometric modeling. Any boolean operation can be seen as an intersection calculation followed by a selection of parts necessary for building the surface of the resulting object. This paper deals with the computing of intersection curves on subdivision surfaces (surfaces generated by the Loop scheme). We present three variants of our algorithm. The first variant calculates this intersection after a classification of the object faces into intersecting and non intersecting pairs of faces. The second variant is based on the 1-neighborhood of the intersecting faces. The third variant uses the concept of bipartite graph.

Keywords

Subdivision surfaces, Loop scheme, intersection curves, bipartite graph.

1. INTRODUCTION

Surface generation methods play a very significant role in Computer graphics and Computer Aided Design (CAD). Subdivision surfaces based on shape modeling has two main advantages: it applies to meshes of arbitrary topology (like polygonal modeling) and it exhibits local behavior (like NURBS or B-Spline modeling); it is also based on only a small number of control points. Subdivision techniques are now widely used. The success of these surfaces is due to their capacity to generate smooth surfaces starting from arbitrary initial meshes and to their relatively easy implementation thanks to their simple concept. A subdivision surface is defined by an initial mesh of arbitrary type and some refinement rules. These rules consist of geometrical rules that determine the positions of new control points from the positions of old ones and topological rules which describe the refinement procedure of the control polyhedron connectivity and thus surface properties. From a polygonal mesh, called the control network, the repeated application of refinement rules produces new polygonal meshes including more and more faces. The produced sequence of meshes converges towards a smooth surface, called the limit surface (e.g. B-spline or Box-spline), topologically similar to the initial control network. Figure 1 shows an example of a subdivision surface. From left to right, the surface becomes increasingly smooth.

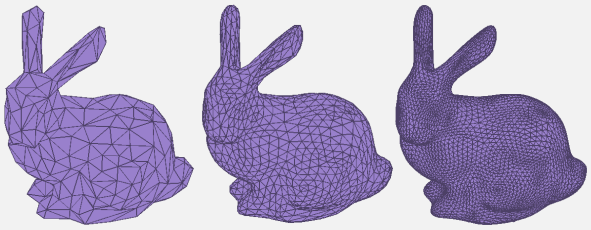

Figure 1. Example of a subdivision surface.

Since the introduction of subdivision surfaces in 1978 by Catmull-Clark [Cat78] and Doo-Sabin [Doo78], many subdivision schemes have been proposed [Loo87, Zor00, Kob00, Vel00].

In geometric and surface modeling, computing surface intersections is a fundamental and recurrent problem widely studied for algebraic and parametric surfaces. Depending on the algorithms involved in the different underlying tasks, intersection computation

WSCG POSTERS proceedings
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

methods may be classified into four main categories [Boe91, Pat93, Abd96]: Analytical methods [Cha87, Pat93], lattice evaluation methods [Bar87], marching methods [Baj88] and subdivision methods.

In the present work, we are particularly interested in computing intersection curves of subdivision surfaces in the context of solid algebra where objects are all modeled by subdivision surfaces. Although for practical reasons our intersection computing algorithm has only been tested on subdivision surfaces generated by the Loop scheme, our method is nonetheless applicable to any other subdivision scheme. In this paper, we present and compare three variants of a subdivision surface intersection computation algorithm: the first variant, called the natural variant, calculates this intersection after having classified the object faces into intersecting or non intersecting pairs. The second variant is based on the 1-neighborhood of the intersecting faces. The third variant uses the concept of bipartite graph. To apply the two last variants, it is necessary to know the intersecting faces and intersection curves computed using the natural variant at the initial level. The main difference between the neighborhood and graph versions lies in the set of faces considered. At a given level of subdivision, the first considers a set of faces per object, while the second considers several subsets of the precedent set by the intermediary of a bipartite graph.

2. NATURAL ALGORITHM

Computing intersection curves is carried out in several steps. First, the faces of two surfaces are indexed into two categories: intersecting faces and others. Only intersecting pairs of faces are considered in the next step of the algorithm. The face/face intersection results in a set of intersection points which will be sorted and connected in order to obtain piecewise linear approximations of the intersection curves.

Detecting face intersection

This preliminary step consists in using the enclosing boxes collision tests as a first filter to eliminate faces clearly disjoined from any future investigation. Then, intersections of all the remaining faces will be calculated. The complexity of this algorithm becomes $O(m_1 \times n_1)$ where m_1 and n_1 respectively are smaller than the original numbers of faces m and n of the surfaces S_1 and S_2.

Computing face intersection

Several methods may be considered to compute intersection points. O'Brien and Manocha [Obr00] compute the intersection by making the intersection of the planes carrying the faces and then by taking the restriction on the faces. They are thus obliged to distinguish between two cases: the case in which the two intersection points belong to the edges of the same face and the case in which they are carried by segments of different faces (Figure 2).

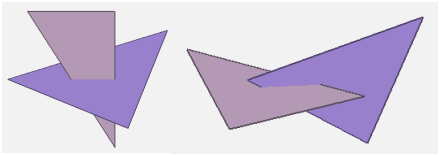

Figure 2. Two cases of face/face intersection.

Another solution consists in computing the intersections of all the segments of a face with the other face. The straight line/plane intersection is calculated first. The equation of the straight line being written in parametric form, the restriction on the segment is carried out by checking that the value of the parameter is in the interval $[0,1]$. Then we need only to check that the intersection point belongs to the face, which is easily done by comparing the surface of the face with the sum of the surfaces of the triangles formed by the intersection point and the face vertices. We chose to implement this second solution.

The polygonal intersection curve

Point sorting is easily carried out thanks to the structure of the intersection point which stores the point coordinates, the faces of each object at the origin of this point and the edge on which this point is located. Thus the ends of the intersection edges correspond to the intersection points pointing to two identical faces. Then, the edges are connected by using the winged edge structure [Bau72]. **Figure 3** represents an example of the intersection curve and the intersecting faces at the initial level.

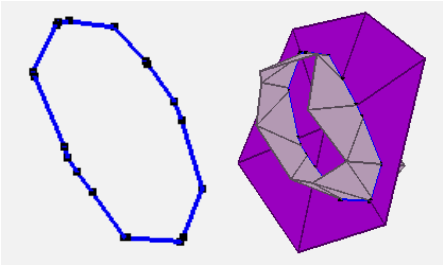

Figure 3. Intersection curve and intersecting faces at the initial level.

3. NEIGHBORHOOD ALGORITHM

This algorithm only considers the faces of the neighborhood of the previous intersecting curves as in [Obr00]. The calculation of the intersection at the initial level for this algorithm is carried out using the natural algorithm. Intersecting pairs of faces of two surfaces are thus determined.

Let I_i be the face set of the surface S_i involved in the intersection.

The first step of this algorithm consists in finding the 1-neighborhood $\mathcal{V}_1(I_i)$. Let us remember that this 1-neighborhood contains the set I_i as well as all faces adjacent to these faces by a vertex. The sets $\mathcal{V}_1(I_1)$ and $\mathcal{V}_1(I_2)$ are represented in the Figure 4.a.

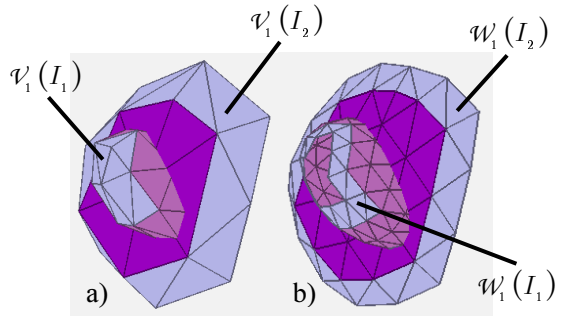

Figure 4. Intersecting faces. a) 1-neighborhood $\mathcal{V}_1(I_i)$. b) 1-neighborhood subdivided $\mathcal{W}_1(I_i)$.

In fact, the 2-neighborhood $\mathcal{V}_2(I_i)$ is nevertheless preserved in order to correctly calculate the 2-neighborhood $\mathcal{W}_2(I_i)$ of the set of sub-faces of I_i obtained by the subdivision of Loop. Indeed, in certain cases, the faces of $\mathcal{W}_2(I_i)$ are required to find the entire neighborhood at the following level. Then, the natural algorithm is again used to test intersections between $\mathcal{W}_1(I_1)$ and $\mathcal{W}_1(I_2)$. The intersection curve obtained at level one is given in Figure 5.

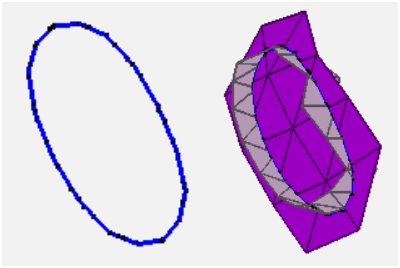

Figure 5. Intersection at level 1. Left: The intersection curve. Right: Intersecting faces

This process will be repeated x times to obtain the intersection curves at a given subdivision level.

This algorithm significantly reduces the number of faces involved in the intersection calculation; it is consequently much faster than the natural algorithm.

4. ALGORITHM USING A BIPARTITE GRAPH

This algorithm uses a bipartite graph to reduce the number of intersections to be tested.

The nodes of one column represent the faces of the first object and those of the second column represent the faces of the second object. Intersecting faces are connected by edges.

Once this graph is established, the neighborhoods of F_i and G_k are added to the graph. The intersections of each face of $\mathcal{V}_1(I_1)$ with all the faces of $\mathcal{V}_1(I_2)$ are no longer tested, as the algorithm consists in computing only intersections between sub-parts of these sets. Indeed, the bipartite graph of intersection associates to each intersecting face F_i of I_1, a set G_i of I_2 intersecting faces, so that one needs only to calculate the intersections between elements of $\mathcal{V}_1(F_i)$ and elements of $\mathcal{V}_1(G_k)$ connected by an edge.

Below, we apply this algorithm to a simple example (to reduce the number of graph vertices): the intersection of the bunny (694 faces) with a smaller band (10 faces) is considered.

Our algorithms is composed by the following steps: construction of the initial level bipartite graph (Figure 6), neighborhood computation (Figure 7), computation of partial subdivision (Figure 8). Finally, the intersecting couples of faces are determinated.

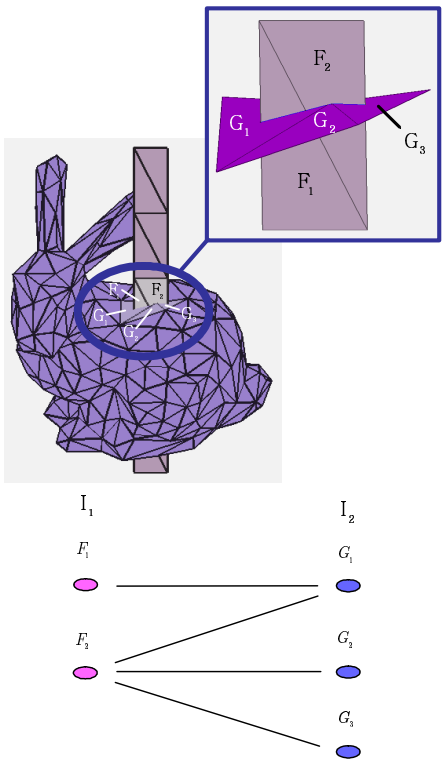

Figure 6. Construction of the initial bipartite graph.

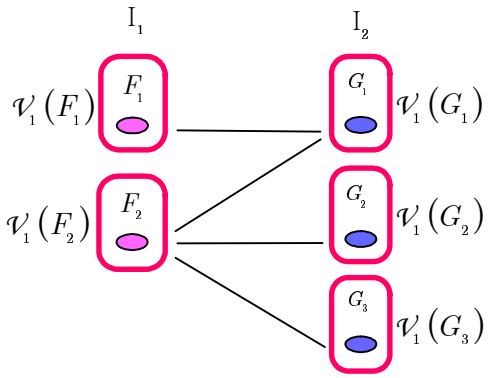

Figure 7. Computation of the 1-neighborhoods $\mathcal{V}_1(F_i)$ and $\mathcal{V}_1(G_k)$.

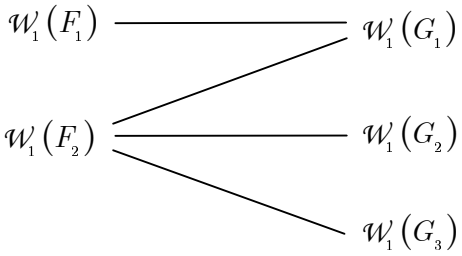

Figure 8. Obtaining $\mathcal{W}_1(F_i)$ and $\mathcal{W}_1(G_k)$ from partial subdivision of $\mathcal{V}_1(F_i)$ and $\mathcal{V}_1(G_k)$.

This variant is more efficient than the first and the second variants presented above. The fact that it is based on a bipartite graph reduces the number of tests for intersecting faces at each subdivision level.

5. CONCLUSION

In this paper, we described three variants of an algorithm for the computation of intersection curves between two objects modeled by subdivision surfaces. The natural algorithm is a non optimized variant that can be interesting to use when the intersecting subdivision surfaces have a small number of faces. The other two variants proposed are respectively based on the concepts of 1-neighborhood and bipartite graph. In particular, determining pairs of intersecting faces of the third algorithm (Section 4) may be improved in order to avoid systematically computing the intersection between $\mathcal{W}_1(F_i)$ and $\mathcal{W}_1(G_k)$ for each $F_i \in I_1$, $G_k \in I_2$ and $(F_i, G_k) \in E$ of G^0. These algorithms compute intersection curves between two objects more quickly than the natural algorithm, in particular when the number of faces involved is very high. It now remains to integrate this improvement into the framework of boolean operations.

6. REFERENCES

[Abd96] K. Abdel-Malek, H. J. Yeh. "Determining intersection curves between surfaces of two solids". Computer Aided Design, vol. 28-6/7, pp 539-549, 1996.

[Baj88] C. L. Bajaj, C. M. Hoffmann, J. E. Hopcroft, R. E. Lynch. "Tracing surface intersections". Computer Aided Geometric Design, vol. 5, pp 285-307, 1988.

[Bar87] R. E. Barnhill, G. Farin, M. Jordan, B. R. Piper. "Surface/surface intersection". Computer Aided Geometric Design, vol. 4-3, pp 3-16, 1987.

[Bau72] Bruce G. Baumgart, "Winged edge polyhedron representation", Technical Report CS-TR-72-320, pp 5, 1972.

[Boe91] E. Boender. "A survey of intersection algorithms for curved surfaces". Computer & Graphics, vol. 15-1, pp 99-115, 1991.

[Cat78] E. Catmull, J. Clark. "Recursively generated B-spline surfaces on arbitrary topological meshes". Computer Aided Design, vol. 9-6, pp 350-355, 1978.

[Cha87] V. Chandru, B. S. Kochar. "Geometric Modeling: Algorithms and NEW Trends". Chapter Analytic Techniques for Geometric Intersection Problems, pp 305-318, SIAM, Philadelphia, PA, 1987.

[Doo78] D. Doo, M. Sabin. "Behaviour of recursive subdivision surfaces near extraordinary points". Computer Aided Design, vol. 9-6, pp 356-360, 1978.

[Kob00] L. Kobbelt. "Sqrt(3)-Subdivision". Computer Graphics Proceedings, Annual Conference Series, pp. 103-112, July 2000.

[Kri94] S. Krishnan, A. Narkhede, D. Manocha. "Boole: A System to Compute boolean Combinations of Sculptured Solids". Technical Report, Department of Computer Science, University of North California, 1994.

[Loo87] C. Loop. "Smooth Subdivision Surfaces Based on Triangles". Master's thesis, University of Utah, Department of Mathematics, 1987.

[Obr00] D. A. O'Brien, D. Manocha. "Calculating Intersection Curve Approximations for Subdivision Surfaces", 2001. http://www.cs.unc.edu/~obrien/courses/comp258/project.html

[Pat93] N. M. Patrikalakis. "Surface-to-surface intersections". IEEE Computer Graphics & Applications, vol. 13-1, pp 89-95, January 1993.

[Vel00] L. Velho, D. Zorin. "4-8 Subdivision". Computer Aided Geometric Design, volume 18-5, pp 397-427, 2000.

[Zor00] D. Zorin. "Subdivision Zoo". SIGGRAPH 2000 Course Notes, Subdivision for Modeling and Animation, Chap. 4, pp 65-98, 2000.