Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Visible region extraction from a sequence of rational Bézier surfaces*

RUAN Xiao-yu[1,2], ZHANG Hui[2], YONG Jun-hai[2]

(*[1]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*)

(*[2]School of Software, Tsinghua University, Beijing 100084, China*)

E-mail: roketruan@gmail.com; zhanghui@tsinghua.edu.cn; yongjh@mail.tsinghua.edu.cn

Received Apr. 3, 2006; revision accepted Apr. 17, 2006

Abstract: A method for computing the visible regions of free-form surfaces is proposed in this paper. Our work is focused on accurately calculating the visible regions of the sequenced rational Bézier surfaces forming a solid model and having coincident edges but no inner-intersection among them. The proposed method calculates the silhouettes of the surfaces without tessellating them into triangle meshes commonly used in previous methods so that arbitrary precision can be obtained. The computed silhouettes of visible surfaces are projected onto a plane orthogonal to the parallel light. Then their spatial relationship is applied to calculate the boundaries of mutual-occlusion regions. As the connectivity of the surfaces on the solid model is taken into account, a surface clustering technique is also employed and the mutual-occlusion calculation is accelerated. Experimental results showed that our method is efficient and robust, and can also handle complex shapes with arbitrary precision.

Key words: Rational Bézier surface, Visibility, Self-occlusion, Mutual-occlusion
doi:10.1631/jzus.2006.A1233 **Document code:** A **CLC number:** TP391

INTRODUCTION

Computing the visible portions of parameterized surfaces is a fundamental problem for Computer Graphics, Computer Aided Design, Computer Vision, and so on. Generally, a common solution to getting the visible region of the parameterized surfaces is to use discrete methods including Z-Buffer (Greene *et al.*, 1993), Hierarchical Occlusion Map (HOM) (Zhang *et al.*, 1997), Ray-Tracing (Toth, 1985), and tessellating the surface with its u and v isoparametric curves (Maghrabi and Griffiths, 1989; Lu, 1998). These methods have been extensively studied in Computer Graphics, Computer Aided Design, and other related fields. However, due to the limit of precision in these methods, errors are introduced in computing the discrete points on the silhouette of surfaces and may even propagate when calculating the mutual-occlusion regions. Therefore these methods are not appropriate for acquiring high precision. On the other hand, most previous literature mainly discussed the algorithms for calculating the visible regions of a single parameterized surface, and neglect the connectivity among several surfaces in a solid model. These drawbacks are further discussed in the following sections.

In this paper we focus on discussing how to calculate the visible portions of solid models formed by rational Bézier surfaces in parallel light. Because the model is a solid one, the surfaces in the sequence only have one side facing the exterior. Our work is focused on calculating the sequenced surfaces' visible regions' silhouettes under the given precision. The

* Project supported by the National Basic Research Program (973) of China (No. 2002CB312106) and the National Natural Science Foundation of China (Nos. 60533070, and 60403047). The third author was supported by the project sponsored by a Foundation for the Author of National Excellent Doctoral Dissertation of China (No. 200342) and a Program for New Century Excellent Talents in University (No. NCET-04-0088), China

main procedure is described below.

(1) We get the surfaces' self-visible types in parallel light. Here the self-visible type means the visibility of a surface neglecting the other surfaces, and the silhouette corresponding to an independent surface is its self-silhouette. These types include completely visible type, partially visible type and completely invisible type. Then, the completely invisible surfaces can be neglected in the following procedures.

(2) We use the self-silhouette equation of the Bézier surface to calculate the self-silhouette of the surface visible region in the parallel light condition.

(3) The spatial relationship of the sequenced surfaces is used to determine the mutual-occlusion relationship, and get the corresponding visible regions.

The rest of this paper is organized as follows. Section 2 discusses the surface's self-visible type in parallel light. In Section 3, we develop a method for calculating the self-silhouette of the surfaces which are completely visible or partially visible. In Section 4, we describe the features of the surface which are mutually-occluded by other surfaces in the solid model, and we give the details of getting the mutual-occlusion regions in the surfaces. Section 5 gives the experimental results and analysis of our algorithm. In the last section, we conclude our work and give hints for potential future work.

DETERMINATION OF BÉZIER SELF-VISIBLE TYPES

This part discusses how to get the visible portions of the solid model composed of sequenced rational Bézier surfaces. Because the model is a solid one, the surfaces only have one side facing the exterior. Thus the sequenced surfaces always have regions whose normals point towards the light direction, and we cannot see them, so there is no need to deal with them in the remaining steps. On the other hand, the model also has regions with normals opposite to the light direction, and are composed of completely visible surfaces and partially visible ones. Also the first thing we should do is to determine the visible type surfaces.

We could use the convex-hull property of the Bézier surface to simplify the visibility determination process. The control net of a Bézier surface is also its convex-hull, with the control points forming triangles with their neighboring points. So the control point normal is the average of its adjacent triangle normals. We calculate the normal $\boldsymbol{n}$ of the control point in the control net. Then the dot product d between $\boldsymbol{n}$ and the light direction $\boldsymbol{l}$ is given by

$$d=\boldsymbol{n}\cdot\boldsymbol{l}. \tag{1}$$

If $d\geq0$, indicating the normal of the control point is towards the light direction, the point is defined invisible; if $d<0$, indicating the control point normal is opposite to the light direction, then we define that the point is visible. Therefore, by inspecting the result's symbol of each control point in the control net, the visible type of a surface is obtained: if they are all non-negative, the surface is invisible; if they are all non-positive, the surface is visible except for all zero condition in which the surface is invisible; the remaining surfaces need further computation described in the next section. Fig.9b shows the surface's self-visibility result of a spring model composed of 96 rational Bézier surfaces. The arrow depicts the light direction. The grey surfaces are completely visible, the dark ones invisible, while the light grey ones are partially visible.

SELF-SILHOUETTE OF RATIONAL BÉZIER SURFACE

After getting the surface's visible type, we should first compute the self-silhouettes of completely visible and partially visible surfaces. In the previous methods (Pop *et al.*, 2001; Kirsanov *et al.*, 2003), the surface is discretized into triangle-mesh. And then, the silhouette of the mesh is computed, with the precision of result depending on the discrete result. Fig.1 shows that the point C is the point on the mesh silhouette, and C' is the point on the silhouette of the original surface. Because there is an approximation error between the mesh and the original surface, the point on the mesh silhouette may not be on the silhouette of the surface, also C and C' may not be coincident either. So these methods are not suitable for high precision.

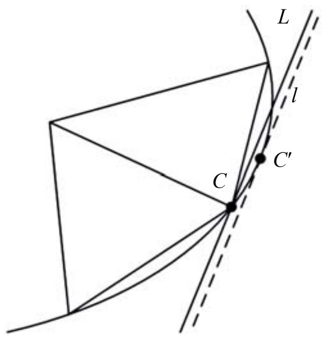

Fig.1 Point C' is on the real silhouette of the surface, and is translated to the point C after the triangulation

Some accurate methods for getting the self-silhouettes of rational Bézier surfaces (Krishnan and Manocha, 2000; Elber and Cohen 1990; 1995) are based on the surface equation, and they guarantee that the discrete points are on the real silhouette. It is obvious that self-silhouettes of completely visible surfaces are boundary curves of the surfaces; self-silhouettes of partially visible surfaces are composed of partial boundary curves and the inner silhouette which is the curve dividing the visible and invisible portions of the surfaces. In Fig.2, the rational Bézier surface is self-occlusive. The dark grey curve represents the self-silhouette on the surface, and the inner silhouette curve is marked with ellipse.

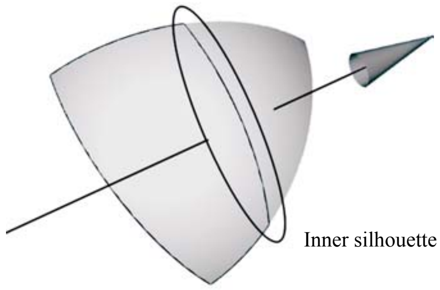

Fig.2 The dark grey curve is the self-silhouette of the rational Bézier surface. The arrow indicates the direction of light, and the curve part marked with ellipse is the inner silhouette

The main work involves getting the inner silhouette. In our approach, we first get the start and end points of the inner silhouette with the silhouette equation. The point on the inner silhouette satisfies

$$\boldsymbol{n}_{\mathrm{s}}(u,v)\cdot\boldsymbol{l}=0, \tag{2}$$

where $\boldsymbol{n}_{\mathrm{s}}(u,v)$ is the normal of the point corresponding to the parameter u and v, and $\boldsymbol{l}$ is the vector of light direction. We could get $\boldsymbol{n}_{\mathrm{s}}(u,v)$ from the equation of the rational Bézier surface (Piegl and Tiller, 1997) as follows:

$$\boldsymbol{S}(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,n}(u)B_{j,m}(v)\omega_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,n}(u)B_{j,m}(v)\omega_{i,j}},$$

where

$$0\le u\le 1,\quad 0\le v\le 1. \tag{3}$$

$\boldsymbol{S}_u(u,v)$ and $\boldsymbol{S}_v(u,v)$ are the partial derivations of u and v. The normal $\boldsymbol{n}_{\mathrm{s}}(u,v)$ of the point in the surface $\boldsymbol{S}(u,v)$ is

$$\boldsymbol{n}_{\mathrm{s}}(u,v)=\boldsymbol{S}_u(u,v)\times\boldsymbol{S}_v(u,v). \tag{4}$$

Thus we could get the equation of the inner silhouette from Eqs.(2) and (4). Because the parameterized surface whose degree is 3 is commonly used in the engineering field, the maximum degree of the sequenced surfaces discussed in this paper is also 3. Ignoring the closed inner silhouettes, we classify the inner silhouettes into 16 kinds according to the positions of start and end points, as shown in Fig.3.

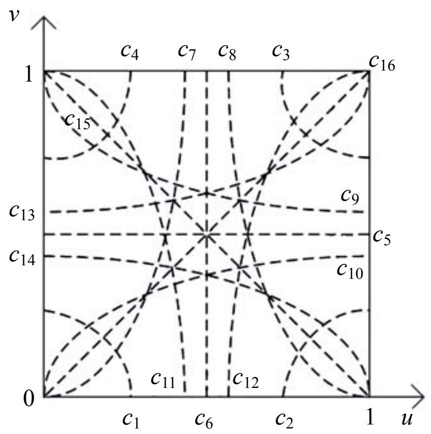

Fig.3 There are 16 kinds of the inner silhouette in the parameterized field. The dash lines indicat the inner silhouette

Table 1 lists the parameter range of start and end points corresponding to 16 kinds of inner silhouettes shown in Fig.3.

Table 1 The types of root corresponding to the inner silhouettes shown in Fig.3

Silhouette	u=0	u=1	v=0	v=1
c_1	(0<v<1)	−	(0<u<1)	−
c_2	−	(0<v<1)	(0<u<1)	−
c_3	−	(0<v<1)	−	(0<u<1)
c_4	(0<v<1)	−	−	(0<u<1)
c_5	(0<v<1)	(0<v<1)	−	−
c_6	−	−	(0<u<1)	(0<u<1)
c_7	(v=0)	−	(u=0)	(0<u<1)
c_8	−	(v=0)	(u=1)	(0<u<1)
c_9	(v=1)	(0<v<1)	−	(u=0)
c_{10}	(v=0)	(0<v<1)	(u=0)	−
c_{11}	(v=1)	−	(0<u<1)	(u=0)
c_{12}	−	(v=1)	(0<u<1)	(u=1)
c_{13}	(0<v<1)	(v=1)	−	(u=1)
c_{14}	(0<v<1)	(v=0)	(u=1)	−
c_{15}	(v=1)	(v=0)	(u=1)	(u=0)
c_{16}	(v=0)	(v=1)	(u=0)	(u=1)

Therefore, we can get the self-visible type for the surface that is not acquired in Section 2. If there is no root on all four boundaries, we say that the surface is completely visible; otherwise, the surface is partially visible.

In the solid model, the sequenced Bézier surfaces are rational ones, the equation of the inner silhouette has variables in the denominator. Therefore, the silhouette equation could have a high degree which may be more than 10. To avoid the instability in solving such a high degree equation, we use the monotonic property of the silhouette curve. So we use the start point (u_0,v_0) and the end one (u_1,v_1) to get the range of u (or v), and we uniformly sample it and solve the corresponding v (or u) of each sample point. Then we get a series of points on the silhouette.

Here, we give an example with the inner silhouette c_5 in Fig.3. The range of the parameter u is [0,1]. We uniformly sample k points $\{u_0, u_1, \ldots, u_{k-1}|u_i=i/(k-1)\}$ in the range. And every parameter v_i corresponding to u_i can be solved from Eq.(2). Because of the uniform sampling of one parameter, we may get great difference between the normal of adjacent points on the surface, which may result in losing the geometry information on the silhouette. So we insert the average parameter $u'=(u_i+u_{i+1})/2$ between u_i and u_{i+1}, and also calculate the corresponding parameter v'. Then we get the new result of the adjacent points and the difference between their normals. Repeat this step until the series of points satisfies the precision requirement.

The self-silhouette of partially visible surface is composed of part of the boundary curves and the inner silhouette. After getting the inner silhouette, we should add proper boundary curves to make the result a loop. Since the inner silhouette is the curve between the visible part and the invisible one of the surface, we could distinguish them by getting the normal of a sample point on one of the two parts and use its parameterized coordinate to get the symbol of Eq.(2). If the symbol is negative, the part, where the sample point locates is visible; otherwise, the part is invisible. Regarding c_5 in Fig.3, we give an illustration on how to get the visible range of a surface. We use Eq.(2) to get the symbol of the result at the parameterized point (0.5,0). If the corresponding symbol is negative, the parametric boundaries of the visible region are $[0,0]$~$[1,0]$, $[1,0]$~$[1,v_1]$, $[1,v_1]$~$[0,v_0]$, and $[0,v_0]$~$[0,0]$; otherwise, the parametric boundaries are $[0,v_0]$~$[1,v_1]$, $[1,v_1]$~$[1,1]$, $[1,1]$~$[0,1]$, and $[0,1]$~$[0,v_0]$. Finally, we get the parametric boundaries of the self-visible region of the surface. Figs.7b, 8b and 9b show some instances of surface self-silhouette shown in thick curves.

MUTUAL OCCLUSION

Surfaces in the sequence may have self-occlusion and mutual-occlusion with other surfaces in the light $\boldsymbol{l}$. Fig.4 shows the mutual-occlusion between the surfaces S_1 and S_2; the light $\boldsymbol{l}$ intersects both S_1 and

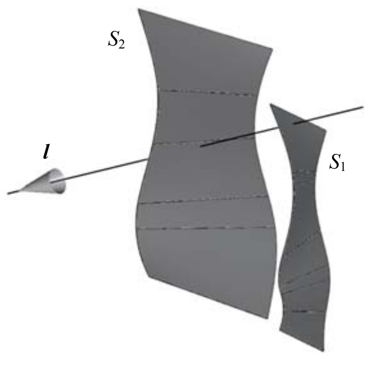

Fig.4 Light $\boldsymbol{l}$ intersects surface S_1 and S_2 respectively

S_2. And we can get the mutual-occlusion region, which is the shadow of S_1 in S_2 when we project the self-silhouette of S_1 onto S_2. But it is not appropriate to perform such work for each pair of surfaces due to the combinatorial complexity resulting from the great quantity of the sequenced surfaces and the complexity of each computation. So we divide the work into two steps: first, find the mutual-occlusion surface pairs using efficient methods; second, we deal with these pairs and get the occluded region in the surfaces. The following subsections give details of these steps.

Mutual-occlusion relationship between the surfaces

Because the spatial relationship determines the mutual-occlusion relationship for each of the two surfaces in the parallel light $\boldsymbol{l}$, we should first get the spatial relationship among the surfaces. The self-silhouette indicates the boundary of the visible region in the light $\boldsymbol{l}$. If we project it onto a plane which is orthogonal to the light, we can get a polygon P. Also the polygons may overlap each other due to the mutual-occlusion of the corresponding surfaces. Fig.5 illustrates the projections P_1 and P_2 of surfaces S_1 and S_2. $S_\perp$ is the projection plane. The curves are the projections of self-silhouettes of the surfaces, and the grey region I refers to the intersection region of the projection of S_1 and S_2.

In this paper, we define the ray-line whose direction is $\boldsymbol{l}$. So we can get a ray-line, which originates from the point p in the grey region I. Then it will intersect the two surfaces at p_1 and p_2, with the intersection points being

$$p_i=p+d_i\cdot\boldsymbol{l}, \qquad i=1, 2. \tag{5}$$

If $d_1>d_2$, the surface S_1 is occluded by S_2; if $d_1<d_2$, the surface S_2 is occluded by S_1.

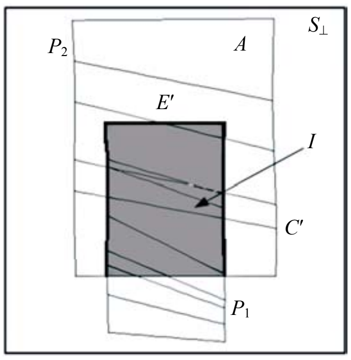

Fig.5 P_1 and P_2 are the projection of the surface S_1 and S_2 respectively. A is the silhouette projection of the visible region of S_2. It is composed of original self-silhouette projection C' and the new boundary projection E' created in the polygon difference operation. The grey region I represents their intersection region

Cluster surfaces

After getting the intersection projection's self-silhouette, we can be sure of the mutual-occlusion relationship between the surfaces. And it is a time consuming process to do the polygon intersection test for every pair of surfaces. We employ a clustering technique to improve the efficiency of the intersection test, and divide all the surfaces, including completely visible and partially visible ones, into different clusters. Then the surfaces within a cluster will have no mutual-occlusion, and any two mutually-occluded surfaces must be laid in different clusters.

Based on the self-visible types of the surfaces obtained in Section 2, we first find the neighboring surfaces for each surface according to the boundary isoparametric curves. Then we cluster the adjacent surfaces which have the same self-visible type. Because there may be mutual-occlusion relationship between partially visible surfaces and their neighbors, we set each one of them in different clusters. So before we determine whether two surfaces are mutually-occluded, we should first test whether they are in the same cluster, so that the efficiency of the intersection test is largely improved, related to the number of clusters. If the number of cluster is small, the number of mutually-occluded surfaces is small as well. Section 5 gives some examples for comparing the results of the time consumed by the algorithm with and without clustering surfaces.

Calculation of the mutual-occlusion region

If we get the silhouette of the mutually-occluded surface, then we get the mutual-occlusion regions and the visible regions. And the points on the silhouette are the intersection points (Lasser, 1986; Barth and Stürzlinger, 1993) of the ray-lines originating from the points composing the boundary of the intersection region I, and the surface. In this way, if we compute all the intersection points and link them, we can get the mutual-occlusion silhouette, and also know the visible region. As a result, the union of the boundaries of all the overlapping regions on a surface is the silhouette of the mutually-occluded region. So the complexity of our method is related to the scale of the mutual-occlusion result and so it is output sensitive.

Conclusion of the overall algorithm

In the foregoing subsection, we give a method to get the mutual-occlusion region by testing the intersection of silhouette projection on a plane perpendicular to the light. Directly applying it requires that each surface must intersect with other surfaces, so the efficiency becomes a problem. In order to improve it, the clustering technique is applied. Also we test the intersection of the bounding box of each projection to eliminate unnecessary polygon intersection tests. In the remaining pairs, the projection may be intersected by several others. So we first use the method (Bentley and Ottmann, 1979; Hobby, 1999) to get the union of their projections, and then use the line-Bézier surface intersection algorithm to get the final result.

In Fig.5, the silhouette projection A of the visible region is composed of the partial self-silhouette projection C' and the new boundary E' created in the polygon difference operation. Thus the points on A are made up of the point c_i' on C' and the point e_j' on E'. Because a surface's self-silhouette and its projection is known, the point on the surface silhouette corresponding to c_i' just requires searching in the point list of the self-silhouette. The point in the surface corresponding to e_j' can be obtained by computing the intersection of the ray-line, originating from e_j', and the surface S. In this way, we get the silhouette of the mutual-occlusion surface.

Part of the silhouette projection may be new boundary. If we project the points onto occluded surface directly, the adjacent points may have great difference between their normals, and then the occluded region's silhouette will be coarse. So we should use an interpolation method similar to that mentioned in Section 3. We insert the mid point $p=(p_i+p_{i+1})/2$ between the adjacent points p_i and p_{i+1}. Then we calculate the intersection of the ray-line, originating from p, and the surface S_2. Repeat the steps until the points in the silhouettes satisfy the precision requirement. Fig.6 shows the result of mutual-occlusion of the surface S_1 and S_2 in the light $\boldsymbol{l}$.

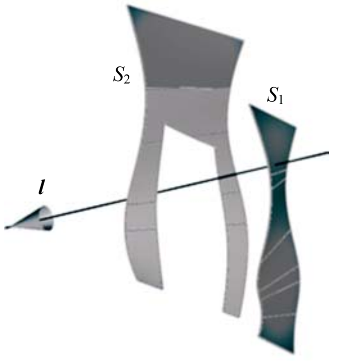

Fig.6 The result of mutual-occlusion of the surface S_1 and S_2 with respect to the light direction $\boldsymbol{l}$

The algorithm of the mutual-occlusion problem is analytically described as follows.

```
1. Project all the self-silhouette points $e$ of the surface onto the plane orthogonal to the light $\boldsymbol{l}$, recording the points coordinate $e$ and the projection coordinate $c$. Then use the maximum and minimum $x$ and $y$ coordinates from the projection coordinate frame, and set the coordinate range to be the bounding box $B$ for the self-silhouette projection;
2. Cluster the surfaces based on comparing the control points of the boundary isoparametric curves;
3. for (each completely visible and partially visible surface surf1) {
4.   for (each other completely visible and partially visible surface surf2 which is behind surf1 in the surface list, and also in different cluster from surf1) {
5.     if (the bounding box $B_1$ intersects $B_2$) {
6.       if (surf1's projection $A_1$ and surf2's projection $A_2$ have intersection $A$) {
7.         Calculate the intersections of the ray-line originating from the point on $A$, and two surfaces surf1 and surf2. And also get their distance $d_1$ and $d_2$
8.         if ($d_1>d_2$)
9.           add surf2 to the occlusion-list of surf1;
             // surf1 is occluded by surf2
10.        else if ($d_1<d_2$)
11.          add surf1 to the occlusion-list of surf2;
             // surf2 is occluded by surf1
       }
     }
   }
 }
12. for (each completely visible and partially visible surface surf) {
13.   if (surf->occlusion-list is not empty) {
14.     Calculate the union $b_u$ of the surf's projection in the shelter-list;
15.     Calculate the result $M$ of the surf's projection $A$ minus $b_u$;
16.     for (each point $m_i$ in $M$) {
17.       if ($m_i$ is equal to the $c_j$ in $A$)
18.         add $e_j$ to the visible region's silhouette of surf;
            // $e_j$ is the point corresponding to the $m_i$ in surf
19.       else    // $c_i$ is not in $A$
20.                 Get the intersection point $a_i$ of the ray-line
                    originating from $c_i$ and following the light di-
                    rection, and the surface surf, also add $a_i$ to the
                    visible region's silhouette of surf;
21.             if (the difference of adjacent point normals does
                    not satisfy the precision requirement)
22.                 add the new silhouette point between them,
                    until the difference of adjacent point normals
                    satisfies the precision requirement;
            }
        }
23.     else        // shelter-list of surf is empty
24.         add the points on the surf's self-silhouette to the
            visible region's silhouette;
    }
```

From the above pseudo code of the algorithm, we can see that the complexity of this algorithm is related to the number of clusters and the number of surfaces in each cluster. In practice, it is related to the distribution of the sequenced surfaces in a certain light direction. Then we get the conclusion that if the number of cluster is large, the number of mutual-occlusion tests is large as well. Also if there is only one cluster, there will be no mutually-occluded surface in the sequence.

EXPERIMENTAL RESULTS

Some examples are presented in this section to illustrate the accuracy and efficiency of the proposed algorithm. Figs.7a, 8a, and 9a are the original models. Figs.7b, 8b and 9b show the results of the models with their self-silhouette. The arrow represents the light

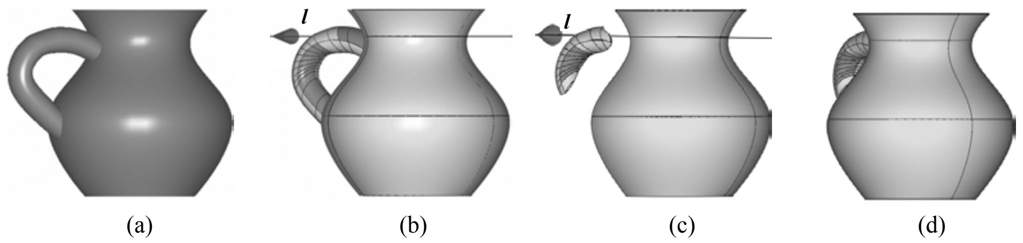

Fig.7 (a) Vase model formed by the surfaces of revolution and sweeping surfaces; (b) Self-silhouettes of the vase's surfaces with respect to the light direction *l*; (c) Trimmed vase model result with respect to the light direction *l*; (d) Visible region of the vase

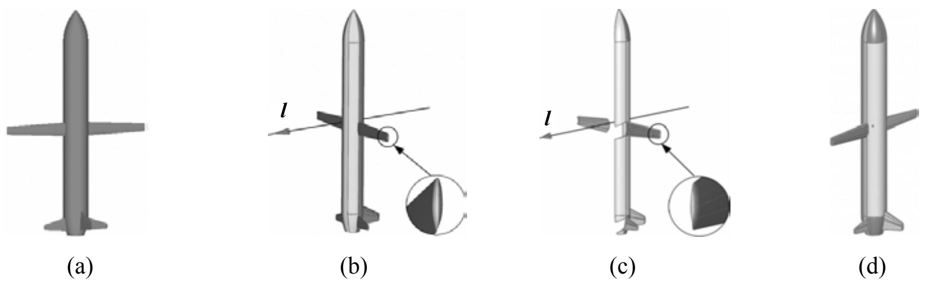

Fig.8 (a) Missile model; (b) Self-silhouettes of the missile's surfaces with respect to the light direction *l*; (c) Trimmed missile result with respect to the light direction *l*; (d) Visible region of the missile

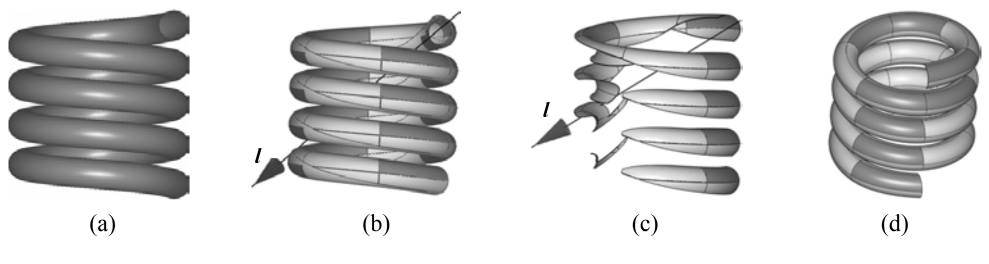

Fig.9 (a) Spring model; (b) Self-silhouettes of the spring's surfaces with respect to the light direction *l*; (c) Trimmed spring result with respect to the light direction *l*; (d) Visible region of the spring

direction ***l***. Surfaces in grey are completely visible in the light direction shown by the arrow, while the partially visible ones are shown in light grey, and invisible ones in dark. Thick curves are the self-silhouettes of the surfaces. Figs.7c, 8c, and 9c show the mutual-occlusion surfaces. Dark surfaces are completely visible, and light grey ones are partially visible, while the invisible regions in the light direction ***l*** are trimmed. Figs.7d, 8d, and 9d show the visible regions of the sequenced surfaces in the light direction.

The time consumed to handle the above models with our algorithm is shown in Table 2. We can see that the effectiveness of clustering procedure is remarkable in the vase model and the missile model. However it has little influence on the spring model, in which the completely visible surfaces and the partially visible ones appear interlaced, so the quantity of clusters is nearly equal to the total number of the two types of surfaces, and the consuming time is not reduced significantly.

Table 2 Comparison of the processing time of the algorithm with and without the clustering procedure

Model	Processing time (s)	
	Without clustering	With clustering
Vase	24.719	7.615
Missile	50.437	22.703
Spring	42.433	42.281

CONCLUSION

Our algorithm is based on the properties of rational Bézier surfaces. Accurate silhouette of the surfaces can be obtained without tessellating them. Therefore arbitrary precision can be obtained. In the process of calculation, the connectivity of the surfaces on the solid model is taken into account, and a surface clustering technique is also employed to improve the efficiency. Application to several complex models demonstrates the robustness and accuracy of the proposed algorithm.

Also our algorithm can be extended to arbitrary surfaces even if they intersect each other. We can divide intersected surfaces into independent patches with the intersection curve, and then add them into the sequence, replacing the original surfaces. Then our algorithm can be applied. In engineering practice, NURBS surface is commonly used. In order to extend our algorithm to it, we can use the Cox-de Boor algorithm (Boehm, 1981) to convert NURBS surfaces to rational Bézier surface patches, and then use our algorithm to deal with them.

References

Barth, W., Stürzlinger, W., 1993. Efficient ray tracing for Bézier and B-spline surfaces. *Computers & Graphics*, **17**(4):423-430. [doi:10.1016/0097-8493(93)90031-4]

Bentley, J.L., Ottmann, T.A., 1979. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, **28**(9):643-647.

Boehm, W., 1981. Generating the Bézier points of B-spline curves and surfaces. *Computer-Aided Design*, **13**(16): 365-366.

Elber, G., Cohen, E., 1990. Hidden curve removal for free form surfaces. *Computer Graphics (SIGGRAPH'90)*, **24**(4):95-104. [doi:10.1145/97880.97890]

Elber, G., Cohen, E., 1995. Arbitrarily Precise Computation of Gauss Maps and Visibility Sets for Freeform Surfaces. Proceedings of the Third ACM Symposium on Solid Modelling and Application, p.271-279.

Greene, N., Kass, M., Miller, G., 1993. Hierarchical Z-Buffer visibility. *Computer Graphics (SIGGRAPH'93)*, **27**:231-238.

Hobby, J.D., 1999. Practical segment intersection with finite precision output. *Computational Geometry: Theory and Applications*, **13**(4):199-214.

Krishnan, S., Manocha, D., 2000. Partitioning trimmed spline surfaces into non-self-occluding regions for visibility Computation. *Graphical Models*, **62**(4):283-307.

Kirsanov, D., Sander, P.V., Gortler, S.J., 2003. Simple Silhouettes for Complex Surfaces. Proceedings Symposium on Geometry Processing, p.102-106.

Lasser, D., 1986. Intersection of parametric surfaces in the Bernstein-Bézier representation. *Computer-Aided Design*, **18**(4):186-192. [doi:10.1016/0010-4485(86)90130-2]

Lu, X.S., 1998. Hidden line elimination for NURBS trimmed surface. *Journal of Computer-Aided Design & Computer Graphics*, **10**(4):315-320 (in Chinese).

Maghrabi, S.M., Griffiths, J.G., 1989. Removal of hidden lines by recursive subdivision. *Computer-Aided Design*, **21**(9):570-576. [doi:10.1016/0010-4485(89)90018-3]

Piegl, L., Tiller, W., 1997. The NURBS book, 2nd Ed. Springer-Verlag, New York, p.50-58.

Pop, M., Huang, W., Barequest, G., Duncan, C., Goodrich, M., Kumar, S., 2001. Efficient Perspective-Accurate Silhouette Computation. ACM Computational Geometry, p.60-68.

Toth, D.L., 1985. On ray tracing parametric surface. *Computer Graphics (SIGGRAPH'85)*, **19**(3):171-179. [doi:10.1145/325165.325233]

Zhang, H., Manocha, D., Hudson, T., Hoff, K.E.III, 1997. Visibility culling using hierarchical occlusion maps. *Computer Graphics (SIGGRAPH'97)*, **31**:77-88.