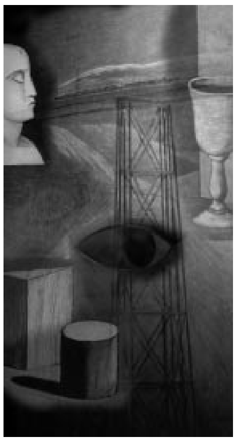
An improved rendering technique for ray tracing Bézier and B-spline surfaces



By Shyue-Wu Wang, Zen-Chung Shih and Ruei-Chuan Chang*

Both numerical and subdivision methods are widely used approaches for ray tracing parametric surfaces. However, the expense of finding the ray–surface intersection points is a major drawback. Thus, simpler and less memory-intensive strategies are needed to improve these methods without further complicating them. This work presents an efficient algorithm for enhancing the performance of both numerical and subdivision methods. The proposed technique can be extended to most applications based on these two methods. The computational time of both approaches is improved by 16–40%. Copyright © 2000 John Wiley & Sons, Ltd.

Received: 20 July 1999; Revised: 12 April 2000

KEY WORDS: ray tracing; Newton's method; Bézier clipping; parametric surfaces; Bézier surfaces

Introduction

Ray tracing parametric surfaces suffers from lengthy computation time, both in calculating the ray–surface intersection points and locating the closest intersection point. To improve performance, the time spent on both parts of the algorithm must be reduced as much as possible. However, researchers seldom consider reducing both factors simultaneously. In this paper, we present an efficient rendering technique for improving numerical methods and subdivision methods by reducing these two factors.

Newton's method is conventionally used to calculate the ray–surface intersection points, and researchers have always focused on how to locate the initial points efficiently. Toth[1] performed interval analysis to find the initial points. Meanwhile, Lischinski and Gonczarowski[2] later proposed an improved technique based on Toth's results. Other researchers[3–8] subdivided surfaces into patches and used various data structures to organize these patches. The initial points are obtained by traveling rays through the data structures and finding the intersections of rays and bounding volumes that enclose the patches. Although these methods[3–8] can accelerate the calculation of the intersection point by selecting suitable initial points, the computational cost of finding the initial points remains high. Joy and Bhetanabhotla[9] first adopted the ray coherence property for selecting initial points. They took the closest intersection point of an adjoining ray previously calculated as the initial point, but faced the problems that an intersection point found through this method might not be the closest one, since not all of the ray–surface intersection points are checked.

Due to the basic process of subdivision methods,[10–12] all the ray–surface intersection points are found by recursively subdividing the surfaces. The closest one is obtained after calculating all of them. That is, subdivision methods do not attempt to reduce the time spent on selecting the closest intersection points.

The ray coherence property is important in accelerating the calculation of the intersection points. For numerical methods, the closest intersection point of an adjoining ray previously calculated can be taken as the initial point for Newton's method. If an intersection point found in this manner can be verified to be the closest, the computational cost of calculating the others can be eliminated.

Ray coherence can also be incorporated into subdivision methods to assist intersection calculation. During subdivision, it is preferable to subdivide the region containing the intersection point found by the previously traced ray. After an intersection point is obtained from this region, whether this point is the closest one is then verified. In this case, the remaining subdividing processes can be avoided. Namely, the

*Correspondence to: Zen-Chung Shih, Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China closest intersection point can be located efficiently without finding the other points. Until now, no researchers have attempted to realize this idea to accelerate subdivision methods.

The following paragraphs describe a rendering technique that combines the ideas of improving both numerical methods and subdivision methods.

Given a surface that will be tested along a scan line, the first ray that intersects the surface along the scan line is located. The closest intersection point is then located and marked. For the next ray, an intersection point is first found by ray coherence and then whether this point is the closest one is verified. If it is the closest point, it is not necessary to calculate all the other intersection points for this ray. Otherwise, all the other intersection points must be found and the closest one selected. This process continues for the succeeding rays along the scan line that intersect the surface.

This rendering technique is advantageous in that it not only reduces time spent on calculating the intersection points but also locates the closest intersection points without finding all the other intersection points. The foundation of this rendering technique is how to verify whether an intersection point is the closest one. Originally, it was planned to follow up the operation of these two methods to modify the process of finding the intersection points without tremendous extra memory storage. With minor adjustment of these two methods to fit the proposed rendering technique, the executive performance could be enhanced. Since numerical and subdivision methods have been applied widely in many practical directions, enhancing performance efficiency with the rendering technique proposed in this paper would be quite appropriate.

Since both numerical and subdivision methods have their own scheme to locate ray–surface intersections, it is difficult to create a general improvement scheme. To verify whether an intersection point is the closest, individual schemes are proposed for numerical and subdivision methods, respectively. Both schemes are based on the basic processes of numerical and subdivision methods, respectively. Two algorithms, a subdivision method and a numerical method, are modified and implemented. Experimental results indicate that the improved algorithms can reduce total rendering time by 16–40%.

The rest of this paper is organized as follows. First, the improvement of numerical methods is described and applied to the algorithm proposed by Barth and Stürzlinger.[4] Then, how to enhance the subdivision method and apply the improvement to Bézier clipping is discussed. Finally, experimental results and discussions are presented.

Numerical Methods

Numerical methods are a suitable and direct way to find ray–surface intersection points. This section focuses on improving the algorithms[3–8] that adopt Newton's method but not utilize the ray coherence property. The algorithm proposed by Barth and Stürzlinger[4] is taken as an example to demonstrate how numerical algorithms can be improved with the novel rendering technique. Meanwhile, the problem of how to verify whether a found intersection point is the closest one should be solved. A scheme based on the subdivision strategy and verification method is suggested, as originally proposed by Barth and Stürzlinger.[4] The objective is to make the novel rendering technique more flexible, suitable and easier to incorporate into numerical methods, because many algorithms[3,5,6] are based on this algorithm.

The following discussion first reviews Barth and Stürzlinger's algorithm. Since this algorithm can deal with both Bézier and B-spline surfaces, Bézier surfaces are selected as an example for explanation. Then, how to enable this algorithm to render an environment with the proposed technique is discussed.

Barth and Stürzlinger's Algorithm

Barth and Stürzlinger's algorithm consists of two steps. In the preprocessing step, the surface is subdivided adaptively into patches until each patch can be approximated well enough using a plane parallelogram. The plane parallelogram is defined by two vectors, $\vec{v}_1$ and $\vec{v}_2$, as illustrated in Figure 1. In each step of the subdivision, a patch is subdivided into halves and the decision whether to halve along the u- or v-axis is based on the curvature in the u- and v-direction. During the subdivision, these patches are arranged in a binary tree. The root encloses the entire surface and the leaves contain the almost plane parts of the surface.

To efficiently detect whether a patch is intersection free, a bounding volume called the *enclosing parallelepiped* is constructed for each patch stored in the binary tree. The enclosing parallelepiped of a patch is spanned by $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$ where $\vec{v}_3$ is obtained from the cross-product of $\vec{v}_1$ and $\vec{v}_2$. Information of the corresponding patch, including the enclosing parallel-

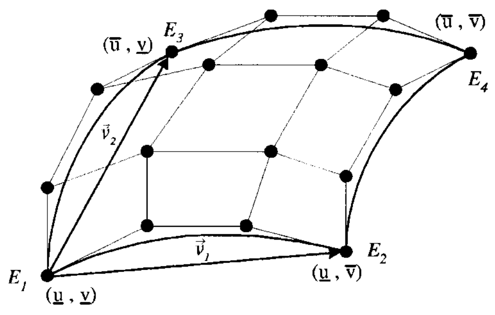

E_3
$(\bar{u}, \underline{v})$
$(\bar{u}, \bar{v})$
E_4
$\vec{v}_2$
$\vec{v}_1$
E_2
E_1
$(\underline{u}, \bar{v})$
$(\underline{u}, \underline{v})$

Figure 1. A part of a Bézier surface. The patch is defined over $[\underline{u}, \bar{u}] \times [\underline{v}, \bar{v}]$.

epiped, is stored in each node of the binary tree, and is employed for the intersection tests or for calculating the initial point. For each leaf node, the parametric domain, and approximating parallelogram are also stored. The following Newton iteration works on the whole surface rather than merely the patch under consideration.

In the rendering step, a ray is traced through the binary tree of a tested surface. The traversal process starts from the root of the tree. If the ray strikes the enclosing parallelepiped, the traversal process continues to the children and the same test is performed again. The above process locates all the leaf nodes whose parallelepipeds are intersected by the ray. For each node, an initial point is generated from the intersection between the ray and the corresponding parallelogram. If Newton's method converges, an intersection point is obtained. Barth and Stürzlinger considered that only one intersection point exists between the ray and the patch. If Newton's method fails to converge, a second iteration process is initiated with the entrance point into the parallelepiped as the initial point. Furthermore, a third iteration will be performed after an unsuccessful second iteration. If these three iteration processes fail to converge, the ray is considered not to intersect the patch.

Attention is also paid to the critical case in which a ray is nearly tangential to the part of the surface. Here, multiple intersections likely exist. This critical case is easily recognized because it occurs only when a ray passes through the parallelepiped approximately parallel to the larger face, as shown in Figure 2. The following inequality is proposed to determine this critical case.

$$\tan\alpha > \frac{b}{h} \qquad (1)$$

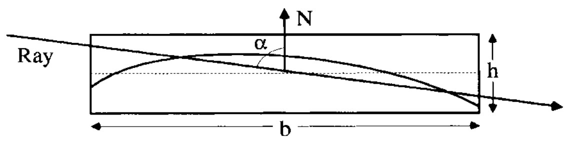


Figure 2. The critical case that a ray is nearly tangential to the part of the surface.

If equation (1) holds, the entrance point of the ray into the parallelepiped is taken as the initial point. Hence, the iteration is *more likely* to converge to the first intersection. However, 'equation (1) does not hold' is not a crucial condition for the patch being free of the intersections, or the located point being the closest one. Some artifacts may appear, but they are not easily noticeable in the anti-aliased picture.

Improvement of Barth and Stürzlinger's Algorithm

Closely examining the basic process of this algorithm reveals that the primary function of the binary tree is to provide an efficient structure for finding the ray–patch pairs. In addition to providing the same capability, regular grids also offer a good mechanism for verifying whether an intersection point is the closet one. Unfortunately, the inherent property of the binary tree cannot provide such a mechanism. Regular grids are used to organize a surface instead of the corresponding binary tree. Details of the modified algorithm are described as follows.

In the preprocessing step, each surface is subdivided according to the rules proposed by Barth and Stürzlinger. A regular grid associated with each surface is then constructed to organize these patches. The regular grid is created by applying the uniform space subdivision technique[13] to the bounding volume of each surface. The bounding volume used herein is an axis–aligned parallelepiped. Although the essential information of each flat patch is stored in the corresponding voxels, the binary tree does not need to be constructed.

In the rendering step, the primary rays are traced through the regular grids of the tested surfaces and the ray–surface intersection points are calculated according to the proposed rendering technique. Given a surface that will be tested along a scan line, the first ray that intersects the surface along the scan line is located. Then the closest intersection point is found and marked. The following ray–surface intersection points are calculated using ray-to-ray coherence along a scan line. Once an intersection point is found via ray coherence, a testing ray is traced from the intersection point toward the origin of the current traced ray. This testing ray is traced to verify whether the intersection point is the closest one by examining the intersection of the testing ray and the visited patches.

While the testing ray is traversing the regular grid, a list of patches whose enclosing parallelepipeds are hit by the testing ray is obtained. Then, equation (1) is checked for the testing ray and the patch containing the intersection point to be verified. If this inequality does not hold, we regard that only one intersection of the testing ray and the patch exists according to the rule proposed by Barth and Stürzlinger. Otherwise, the entrance point of the testing ray into the parallelepiped is used as the initial point for iteration, and a closest intersection point is sought.

Next, if no other patches exist in the list, the verified intersection point is regarded as the closest one. Otherwise, Newton's method is employed to find the intersection points with the initial points obtained from the intersections of the testing ray and the intersected parallelepipeds. The detection process can either determine whether an intersection point is the closest or locate the real closest intersection point for the current traced ray.

The details of this process are illustrated with an example in Figure 3, where the dotted grids represent the regular grid and the curved segments represent the patches. Assume that P_1 is the intersection point found by Newton's method without applying ray coherence and P_2 is the point found using P_1 as the initial point. We conclude that P_2 is the closest intersection point

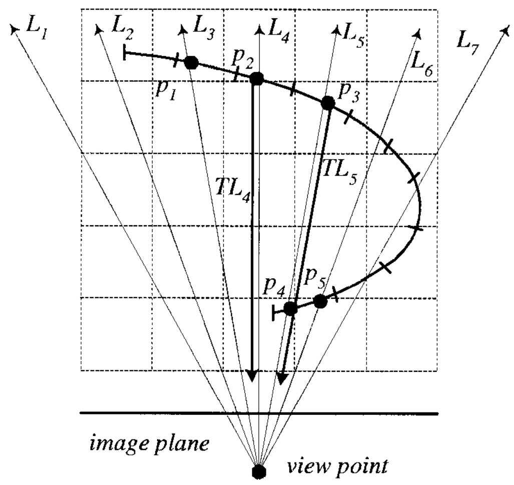


Figure 3. An example of verifying the closest intersection point.

because the testing ray TL_4 only intersects with the patch containing the point P_2. Regardless, if P_3 is the intersection point found with P_2 as the initial point, the closest intersection point P_4 will be obtained, because the testing ray TL_5 will intersect with the patch containing the point P_4.

The above improvement has two advantages. First, this improvement reduces the time spent on both finding the initial points and locating the closest intersection point. Second, this improvement can easily be extended to the previous algorithms.[3,5,6] For example, to render triangular trimmed free-form surfaces, surfaces can be subdivided and the bounding volumes (*tripipeds*) for patches calculated using the methods proposed by Stürzlinger.[6] Then, all patches are stored in the corresponding regular grids. The improved rendering procedure mentioned above can be used to locate the ray–surface intersection points.

Subdivision Methods

Now we discuss the improvement of subdivision methods. We first review the concept of subdivision methods. Most subdivision methods are based on a variation of the same idea. The domains that may contain intersection points are subdivided recursively and those without solutions are pruned. However, each method has a different means of finding the ray–surface intersection points. In this section, Bézier clipping is chosen as the method to be improved to illustrate how almost all the subdivision methods can adopt the proposed rendering technique to trace primary rays.

Compared with numerical methods, subdivision methods are not easily able to exploit ray coherence. The detailed processes of subdivision methods must be analyzed to achieve this goal. This section first details the clipping process of Bézier clipping, and then discusses how to improve Bézier clipping.

Clipping Process of Bézier Clipping

The clipping process[10] for finding the ray–surface intersection points can be classified into two types: *straight-forward* and *bisection*. For the straight-forward case, the u and v axes are clipped alternatively, each step achieves a considerable processing of void parts, and exits with 'no intersections' or 'one intersection'. In Figure 4, u^j (v^j) represents that the jth clipping stage is

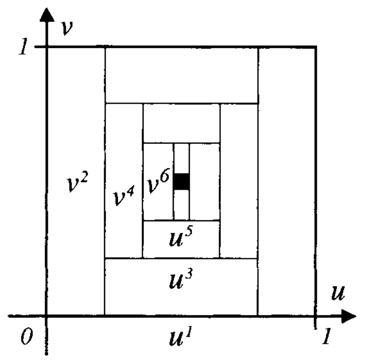


Figure 4. The clipping process of the straight-forward case on finding the ray–surface intersection point.

performed along the u axis (v axis). For example, u^1 denotes that the u axis is clipped at step 1 and v^4 denotes that the v axis is clipped at step 4. In this example, after six steps of clippings, the intersection point is obtained. Of course, this simple method cannot work if there is more than one intersection point in the patch.

For a bisection case, consider Figure 5, where Figure 5(a) illustrates the third clipping step (termed the *bisection step*). The clipping process fails to reduce the interval width of parameter u to a given threshold. Nishita *et al.*[10] suggested a threshold value of 20% of the interval width. Then the parametric space is split into halves and the clipping process is performed on each half. These two parts are denoted herein as u_{low} and u_{high} as illustrated in Figure 5(a). The clipping process is first performed on u_{low}. Then an intersection point is found. On dealing with the u_{high} part, the clipping process also fails to reduce the interval width of parameter v to a given threshold, as illustrated in Figure 5(b). Hence, the u_{high} part is again divided into halves, $u_{high}v_{low}$ and $u_{high}v_{high}$. After recursively following several steps, the other two intersection points are obtained, as presented in Figure 5(c).

An Improvement of Bézier Clipping

The characteristic feature of Bézier clipping, as introduced in the above section, is that the domains generated from the bisection steps are processed randomly and all the ray–surface intersection points are found to obtain the closest one. To improve Bézier clipping in accordance with the rendering technique, the aim herein is to clip the domains generated from the bisection steps in a reasonable order based on the ray coherence property. In this order, it is preferable to deal with the domain that contains or neighbors on the parameters of the closest intersection point found by the previously traced ray. After an intersection point is obtained from some domain, whether this point is the closest one will be verified. If it is the closest, computations of clipping the remaining domains do not need to be performed. Namely, the closest intersection point can be obtained without finding all the others. The following discussion first describes a procedure for verifying whether an intersection point is the closest one. Details of the modified algorithm are given as well.

In fact, while improving Barth and Stürzlinger's algorithm, a procedure for either verifying whether an intersection point is the closest one or locating the real closest one has already been proposed. In this paper, the aim herein is to enable the subdivision methods to solve the verifying problem in accordance with the algorithms themselves. Without introducing further complex methodology to help, the action that uses

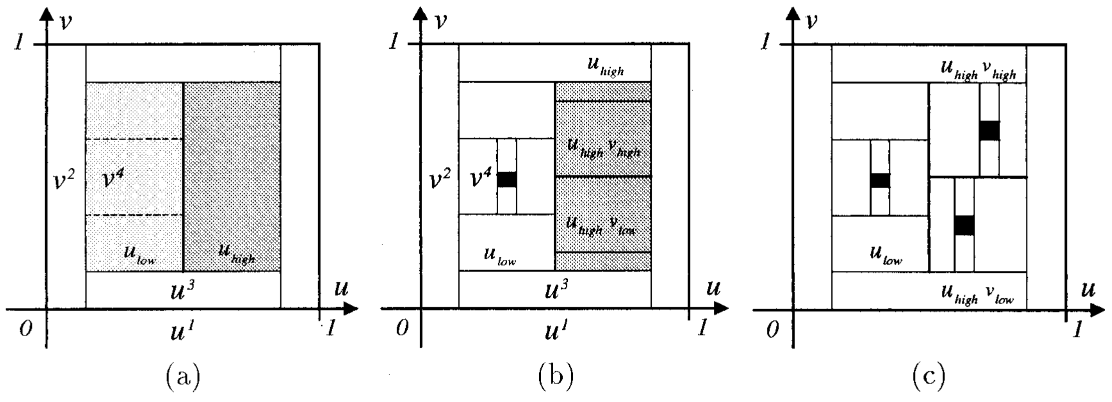


Figure 5. The clipping process of a bisection case.

Newton's method to locate the intersection points is removed from the procedure and make the resulting procedure answer 'yes' or 'no' only. Details of the modified procedure are stated below.

On verifying whether an intersection point is the closest, a testing ray is traced through the point toward the viewpoint as usual. This process obtains a list of patches whose enclosing parallelepipeds are hit by the testing ray. The patch containing the intersection point is first located and equation (1) is then checked for the testing ray and the patch. If this inequality holds, it is reported that this intersection point may not be the closest one, because more than one intersection is very likely to exist. Otherwise, the verification process is continued for other patches of the list. If no other patches exist in the list, it is reported that this point is the closest. If other patches do exist in the list, it is reported that this point may not be the closest one.

The improved algorithm consists of two steps: the preprocessing step and rendering step. In the preprocessing step, the regular grid used to improve Barth and Stürzlinger's algorithm is naturally constructed for each surface. In the rendering step, the primary rays are traced in the scan line order according to the proposed rendering technique. Figure 6 illustrates the process in detail. The dashed lines denote shooting rays from the view point. The black dots indicate that the traced ray intersects the surface S and the closest intersection point is found. B denotes that the closest intersection point is found by direct Bézier clipping. B_1 denotes that the clipping process is the *straight-forward* case and B_m represents that the clipping process is the *bisection* case.

Given a surface that will be tested along a scan line, the ray (the 3rd ray in Figure 6) that intersects with the surface S along the scan line is first located, and then the closest intersection point is found by direct Bézier clipping. For the succeeding rays that intersect with the surface S, if the ray–surface intersection is the straight-forward case, for example, the 4th, 5th, and 6th rays in Figure 6, Bézier clipping is applied to locate the intersection point directly.

For bisection cases, for example, the 7th or 8th rays in Figure 6, the bisection steps will produce some regions. First, the region that contains parameters of the closest intersection point found by the ray previously traced is dealt with; this region is called the *prior region.* Once the first intersection point is found in this region, whether or not this point is the closest one can be verified. If this point is the closest one, the computation is stopped. Otherwise, Bézier clipping is used to find the remaining intersection points.

If no intersections could be obtained from the prior region, dealing with the region that adjoins the prior region is preferable. That is, the parameter domains of this region are beside those of the prior region because the intersection point obtained from the region tends to be the closest intersection point. This process is illustrated with an example shown in Figure 7. No intersection points are obtained from the prior region, hence the clipping process is applied to region R_1 and the intersection point is found, indicated by a black dot. This intersection point is then verified. If this point is the closest, the computation is stopped. Otherwise, Bézier clipping is used to find the other intersection points.

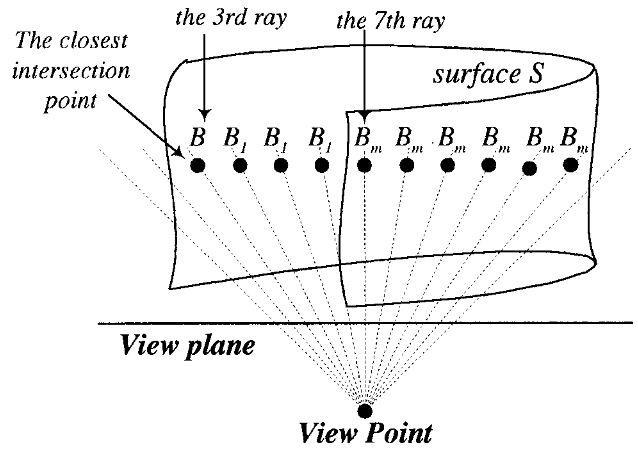


Figure 6. The process of the modified Bézier clipping on finding the ray–surface intersection points along a scan line. The dash lines represent shooting rays from the view point.

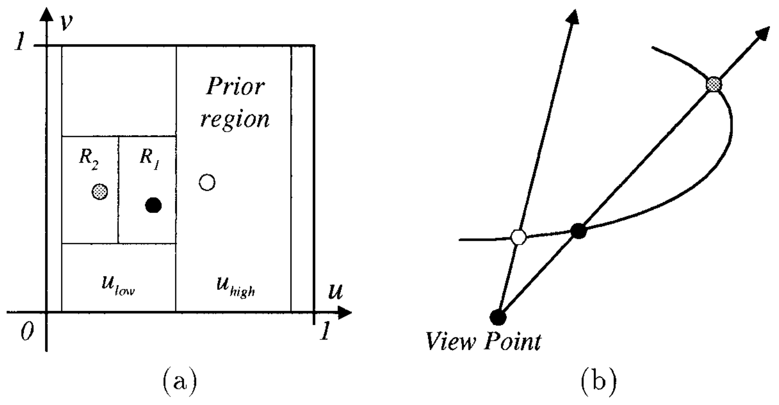


Figure 7. An example of finding the intersections from the non-prior regions. (a) White dot represents the parameters of the closest intersection point found by the previous ray. Black and gray dots represent the parameters of the intersection points between the current ray and the surface. (b) We show the corresponding intersections between two adjoining rays and a surface.

During verification, an alternative approach exists. That is, Bézier clipping can be used to locate the ray–patch intersection points directly rather than reporting that the intersection point is not the closest. In the present implementation it is found that the performance of applying this approach could accelerate restricted performance with the cost of extensive extra memory space to store patches. Hence, this approach is not adopted to improve Bézier clipping.

The major contributions of this improvement are that the number of subdividing processes can not only be reduced, but also that the closest intersection point can be located without finding all of the ray–surface intersection points.

Experimental Results

This section discusses the implementation results and compares them with previous approaches. The proposed algorithm and traditional algorithms are implemented in C programming language. All the results are run on the platform of a Pentium II 300 MHz CPU and 128 MB RAM under a Windows NT 4.0 environment. Photographs for four different scenes were created, Figures 8–11, for performance comparison. The resolution of each image is 1024×1024 and all of the scenes displayed in this paper are generated using anti-aliasing (four rays per pixel) and a ray depth of two. The spectral sampling approach[14] is also employed to produce more realistic color in metals. The implementation herein uses nine spectral samples. Moreover, the flat surfaces, such as a floor or a flat mirror, displayed in this work are created by polygons rather than parametric surfaces.

Uniform space subdivision[13] and simple *min–max* bounding volume (axis-aligned parallelepiped) are used to reduce unnecessary ray–surface intersection tests. Tables 1 and 2 display the results from performing the proposed approach and traditional algorithms on rendering all four distinct scenes. The data included are the sum of rendering a whole scene (not including the shading time) with 512×512 resolutions. Attention is only paid to the total execution time involved in finding ray–surface intersection points. The time units are in seconds.

Performance Comparison of Barth and Stürzlinger's Algorithm

On implementing Barth and Stürzlinger's algorithm, the ray–surface intersection points on the two-dimensional space are calculated. First, a Bézier surface is projected according to two orthogonal planes representing the traced ray. The projection process is identical to that of Bézier clipping.[10] Then Newton's method is applied to solve the two-dimensional non-linear system. For B-spline surfaces, the method proposed by Yang[8] could be used for the projection.

Table 1 presents the total rendering time and total number of intersection points found with the improved Barth and Stürzlinger's algorithm and original algorithm on tracing the primary rays. According to the

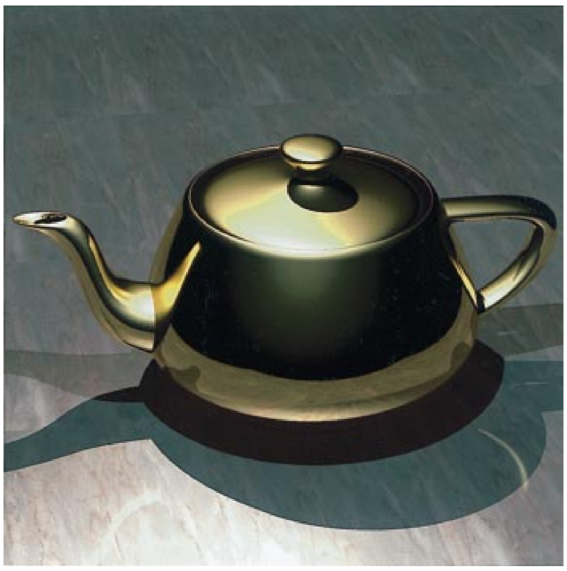


Figure 8. Newell's teapot with 32 surfaces.

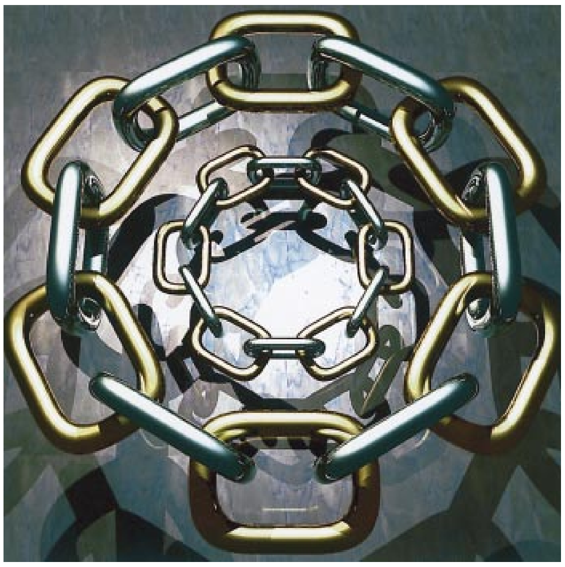


Figure 10. Twenty-four rings with 288 surfaces. Each ring is constructed by 12 Bézier surfaces.

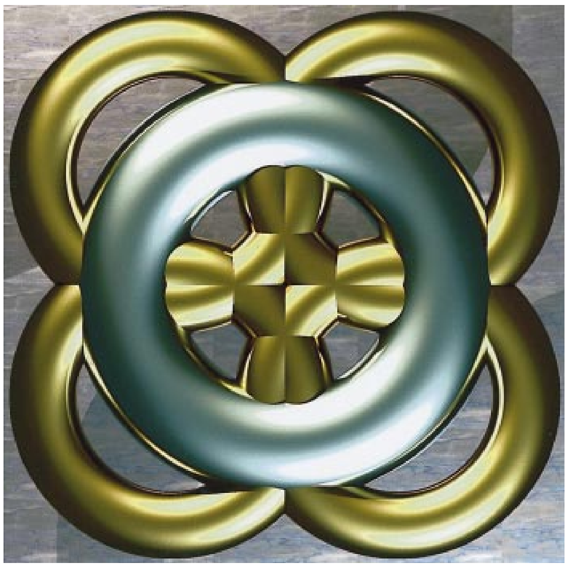


Figure 9. Five toruses with 40 surfaces. Each torus is created by 8 Bézier surfaces.

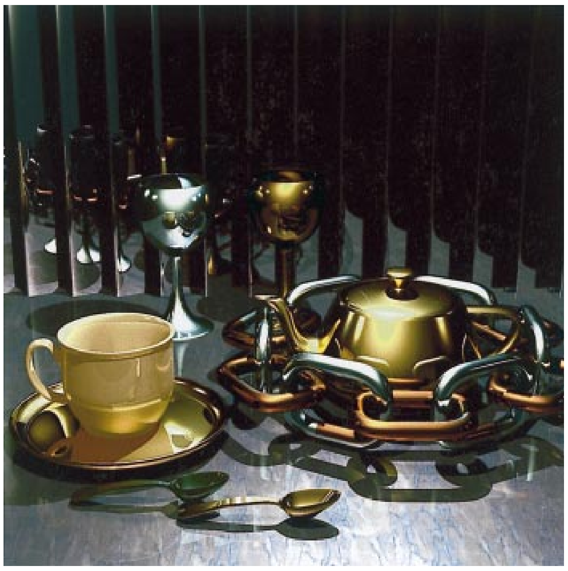


Figure 11. A highly reflective and complex environment with 340 surfaces. The backdrop of the scene consists of curved mirrors. Two mirrors are perpendicular to the floor and located on opposite sides of the floor, which cannot be seen in this rendering view.

experimental results, the proposed method improved the performance by 16–40%. The more high-curvature surfaces that a scene contains, the more computation time is saved. This phenomenon occurs because high-curvature surfaces induce more multiple intersections between the ray and surface. For example, in Figures 9 and 10, most of the rays intersect the surfaces at two intersection points.

Performance Comparison of Bézier Clipping

Table 2 summarizes experimental results of the improved Bézier clipping and original Bézier clipping

Barth and Stürzlinger's algorithm

		Rendering time			No. of intersection points	
Figure	No. of patches/ No. of surfaces	Original algorithm	Improved algorithm	Improved performance	Original algorithm	Improved algorithm
8	7594/32	10.7	8.57	20%	110169	100337
9	13872/40	36.8	21.9	40%	583972	302183
10	39042/288	16.7	13.3	20%	207517	157295
11	39358/340	31.5	26.5	16%	470133	260384

Table 1. Experimental results of the improved Barth and Stürzlinger's algorithm and original algorithm

Bézier clipping

		Rendering time			No. of intersection points	
Figure	No. of patches/ No. of surfaces	Original algorithm	Improved algorithm	Improved performance	Original algorithm	Improved algorithm
8	7594/32	32.9	30.8	6%	146910	102677
9	13872/40	128	101	21%	585952	329157
10	39042/288	66.5	50.3	24%	439421	198208
11	39358/340	145	105	28%	1170938	556281

Table 2. Experimental results of the improved Bézier clipping and original algorithm

on rendering four distinct scenes for the primary ray. Notably, in the present implementation, Bézier clipping was modified according to the schemes suggested by Campagna *et al.*[15] According to the experimental results, the proposed method improved the performance of Bézier clipping by 6–28%. The unsatisfactory performance in rendering Figure 8 is due to the body of the teapot. Most of the intersections between the body and traced rays are the straight-forward case. The execution efficiency can be obviously revealed from the number of intersection points found by the proposed method. The larger the number of intersection points implies a more efficient proposed method.

Comparing Tables 1 and 2 reveals two interesting results. The first result is that Barth and Stürzlinger's algorithm finds the ray–surface intersection points faster than Bézier clipping. This phenomenon occurs because the computation cost of finding an intersection point of Newton's method is potentially lower than that of Bézier clipping. It must be emphasized that this paper aims to propose a rendering technique for improving these algorithms, not to compare the difference and superiority of these algorithms. Each of these algorithms has its own advantages in finding the ray–surface intersection points.

The second result is that the total number of intersection points found by Barth and Stürzlinger's algorithm and Bézier clipping differ from each other, particularly on rendering Figures 9 and 10. This phenomenon occurs because if a ray is (or nearly) tangential to the part of a surface, Bézier clipping will locate many intersection points that are close to each other. This phenomenon occurs in many algorithms, such as the eigenvalue computation or Barth and Stürzlinger's algorithm[3].

The Performance of Nonuniform Sampling

According to the experimental results, the proposed rendering technique works well in the case in which an image plane is uniformly sampled. However, nonuniform sampling also plays an important role in many

Improved Barth and Stürzlinger's algorithm				
		Rendering time		
Figure	No. of patches/ No. of surfaces	Uniform sampling	Jittered sampling	Performance
8	7594/32	33.9	34.7	−1%
9	13872/40	86.5	88.3	−2%
10	39042/288	52.6	53.1	−1%
11	39358/340	102.4	104.5	−2%

Table 3. Experimental results of the improved Barth and Stürzlinger's algorithm

applications, such as anti-aliasing,[18] parallel[16] and progressive[17] ray tracing. To demonstrate the capability of the proposed rendering technique to deal with nonuniform sampling, these four scenes are rendered with a jittered sampling[19] (four rays per pixel) In our implementation, each pixel is subdivided into four subpixels and each sample point is located randomly within its subpixel. Table 3 displays the experimental results from the performance of the improved Barth and Sturzlinger's algorithm.

The above finding indicates that the performance of rendering the jittered sampling is only 2% less than that of rendering the uniform sampling. The same tests were also performed for the improved Bézier clipping, with similar results.

However, the rendering technique proposed herein may be inappropriate for distributed sampling, such as parallel and progressive ray tracing. For parallel ray tracing, the samples are selected adaptively from the image plane, which is based on the balanced load distribution. Furthermore, progressive ray tracing generates the sample location in an order that images may be reconstructed from these samples. The adaptive stochastic sampling of the image plane that the algorithms adopted causes the proposed rendering technique to spend more time on finding the ray–surface intersection points. A similar problem is faced when dealing with the shadow rays, because shadow rays are lack of ray coherence. In our implementation, the improvement of performance for shadow rays is insignificant. Nevertheless, the shadow map[19] is a conventional means of accelerating shadow testing, and can also be applied to ray tracing parametric surfaces. The notion of constructing a shadow map is similar to that of rendering an image plane. Hence, the proposed approach is appropriate for generating shadow maps to accelerate shadow testing.

Conclusions

This paper presents a rendering technique for improving both numerical methods and subdivision methods on rendering Bézier surfaces and B-spline surfaces. Our major contribution is that individual schemes are designed for both the numerical and subdivision methods to find the ray–surface intersection points and locate the closest intersection points efficiently. Simplicity, reduced memory requirements, and enhanced execution are the main characteristics of the proposed rendering technique. The results demonstrate that the improved algorithms can reduce total rendering time by 16–40%. The present implementation only deals with the primary rays. However, the rendering technique can easily be extended to deal with secondary rays by applying the cone tracing[20] or beam tracing[21] technique. In these techniques, rays are traced in a group rather than scan line order.

References

1. Toth D. On ray tracing parametric surfaces. *Computer Graphics (SIGGRAPH '85 Proceedings)* 1985; **19**(3): 171–179.
2. Lischinski D, Gonczarowski J. Improved techniques for ray tracing parametric surfaces. *The Visual Computer* 1990; **6**(3): 134–152.
3. Barth W, Lieger R, Schindler M. Ray tracing general parametric surface using interval arithmetic. *The Visual Computer* 1994; **10**(7): 363–371.
4. Barth W, Stürzlinger W. Efficient ray tracing for Bézier and B-spline surfaces. *Computer & Graphics* 1993; **17**(4): 423–430.
5. Qin K, Gong M, Guan Y, Wang W. A new method for speeding ray tracing NURBS surfaces. *Computer & Graphics* 1997; **21**(5): 577–586.
6. Stürzlinger W. Ray-tracing triangular trimmed free-form

surfaces. *IEEE Transactions on Visualization and Computer Graphics* 1998; **4**(3): 202–214.
7. Sweeney MAJ, Bartels RH. Ray tracing free-form B-spline surface. *IEEE Computer & Graphics Application* 1986; **6**(2): 41–49.
8. Yang CG. On speeding up ray tracing of B-spline surfaces. *Computer Aided Design* 1987; **19**(3): 122–130.
9. Joy K, Bhetanabhotla M. Ray tracing parametric surface patches utilizing numerical techniques and ray coherence. *Computer Graphics (SIGGRAPH '86 Proceedings)* 1986; **20**(4): 279–285.
10. Nishita T, Sederberg TW, Kakimoto M. Ray tracing trimmed rational surface patches. *Computer Graphics (SIGGRAPH '90 Proceedings)* 1990; **24**(4): 337–345.
11. Whitted T. An improved illumination model for shaded display. *Communications of the ACM* 1980; **23**(6): 343–349.
12. Woodward C. Ray tracing parametric surfaces by subdivision in viewing plane. In *Theory and Practice of Geometric Modeling*, W. Strasser and H.-P. Seidel, (ed.); Spring-Verlag: New York; 273–290.
13. Cleary JG, Wyvill G. Analysis of an algorithm for fast ray tracing using uniform space subdivision. *The Visual Computer* 1988; **4**(2): 65–83.
14. Hall R. *Illumination and Color in Computer Generated Imagery*. Springer-Verlag: New York, 1988.
15. Campagna S, Slusallek P, Seidel H. Ray tracing of spline surfaces, bézier clipping, chebyshev boxing, and bounding volume hierarchy: a critical comparison with new results. *The Visual Computer* 1997; **13**(6): 265–282.
16. Notkin I, Gotsman C. Parallel progressive ray-tracing. *Computer Graphics Forum* 1997; **16**(1): 43–55.
17. Painter J, Sloan K. Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics (SIGGRAPH '89 Proceedings)* 1989; **23**(3): 281–287.
18. Glassner AS. *An introduction to Ray Tracing*. Academic Press, California, 1987.
19. Watt A, Watt M. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley: New York.
20. Amanatides J. Ray tracing with cones. *Computer Graphics (SIGGRAPH '84 Proceedings)* 1984; **18**(3): 129–135.
21. Heckbert PS, Hanrahan P. Beam tracing polygonal objects. *Computer Graphics (SIGGRAPH '84 Proceedings)* 1984; **18**(3): 119–127.

Authors' biographies:

Shyue-Wu Wang received the M.S. degree in Computer Science from National Tsing Hua University, Taiwan, R.O.C., in 1992, and is currently a doctoral candidate at National Chiao-Tung University. His research interests include global illumination, real-time rendering, image-based rendering and geometric modeling.

Zen-Chung Shih was born on 10th February 1959, in Taipei, Taiwan, Republic of China. He received his B.S. degree in Computer Science from Chung-Yuan Christian University in 1980, M.S. degree in 1982 and Ph.D. degree in 1985 in Computer Science from the National Tsing Hua University. Currently, he is a professor in the Department of Computer and Information Science at the National Chiao Tung University in Hsinchu. His current research interests include procedural texture synthesis, non-photorealistic rendering, global illumination, and virtual reality.

Ruei-Chuan Chang was born on 30th January 1958, in Keelung, Taiwan, Republic of China. He received his B.S. degree in 1979, M.S. degree in 1981, and Ph.D. degree in 1984, all in Computer Science from the National Chiao Tung University. Currently he is a professor in the Department of Computer and Information Science at National Chiao Tung University in Hsinchu. He is also an associate research fellow at the Institute of Information Science, Academia Sinica, Taipei. His current research interests include design and analysis of algorithms, computer graphics, and system software.