The Visual Computer

A scanline algorithm for displaying trimmed surfaces by using Bézier clipping

Tomoyuki Nishita[1], Kazufumi Kaneda[2], and Eihachiro Nakamae[2]

[1] Fukuyama University, Higashimura-cho, Fukuyama, 729-02 Japan
[2] Hiroshima University, Saijo-cho, Higashi-hiroshima, 724 Japan

Displaying objects with high accuracy is necessary for CAGD (computer-aided geometric design) and for the synthesis of photo-realistic images. Traditionally, polygonal approximation methods have been employed to display free-form surfaces. They bring on low accuracy of display not only in shape, but also in intensity of objects. In this paper, a scanline algorithm to directly display parametric surface patches, expressed by trimmed Bézier surfaces, without polygonal approximation is proposed. In the method proposed here, curved surfaces are subdivided into subpatches with curved edges intersecting with a scanline, and the intersections of every subpatch and the scanline are calculated. This method is extremely robust for calculating the intersections, which can be obtained with only a few iterations; the Bézier clipping method is used for the iteration. Anti-aliased images with shadows and texture mapping are given to show the effectiveness of the method proposed.

Key words: Bézier surfaces – Scanline algorithm – Robustness – High-quality rendering – Surface trimming – Silhouette detection – Shadowing

Offprint requests to: E. Nakamae

1 Introduction

Traditionally, polygonal approximation methods have been employed to display curved surfaces. They can save calculation time and be easily implemented, but the displayed shape is not so accurate to a defined curved surface. To solve these problems, some algorithms rendering bicubic surfaces directly from parametric description have been proposed. One of the authors also developed a raytracing algorithm for trimmed rational surfaces (Nishita et al. 1990), in which Bézier clipping for ray/surface intersection was proposed. The raytracing algorithm is a useful tool for rendering realistic images, but requires expensive calculation time, while scanline algorithms usually can save calculation time. This paper proposes a scanline algorithm for parametric surfaces by using Bézier clipping, which is iterative and based upon the Bézier curve's convex hull; i.e., the intervals of solutions of a polynomial function expressed by the Bézier curve are reduced by clipping the curve, and the process iterates until it converges upon the solutions (Sederberg and Nishita 1990). This method is more robust than Newton's method, which requires determination of a suitable initial value, is not robust, and is difficult to guarantee finding all solutions. Recently trimmed surfaces have become popular in computer-aided geometric design; in CSG (Constructive Solid Geometry) models trimmed surfaces play an important role. The method proposed here can treat trimmed surfaces, too. This paper also deals with penetrated surfaces.

Important elements for rendering curved surfaces are robustness (in the detection of silhouettes), accuracy, required memory, image quality (i.e., anti-aliasing and shadows), and calculation time. In all the above-mentioned methods, there are some problems, such as lack of robustness, in accuray caused by the approximations, and large memory caused by a priori subdivision. Our proposal overcomes all these problems.

The proposed method, although it is partially based on Lane-Carpenter's method (Lane et al. 1980) has several advantages. (a) Subpatches on a scanline are effectively obtained, and no gaps arise between the subpatches. (b) Proposed root finder using Bézier clipping is accurate and robust for calculation of scanline and subpatch intersections. (c) This method is especially effective for displaying scenes with a number of patches, because only the curved surfaces intersecting with the active scanline are subdivided. (d) Anti-aliased images with shadows can be rendered.

The Visual Computer (1991) 7:269–279

2 Previous work

In this section, the previous work of scanline algorithms for parametric surfaces are reviewed.

Blinn (1978) and Whitted (1978) employed the Newton-Raphson method to calculate the intersections of a scanline and curved surfaces. This method needs an initial guess and is not robust. Moreover, a weakness in Whitted's approach is the lack of generality in finding silhouettes. To find silhouettes robustly, Schweitzer and Cobb (1982) proposed a method of dividing curved surfaces into polygons consisting of the boundary curves, which are monotonic in y. In this method, extraction of silhouettes is fairly complicated, because several points on a silhouette need to be calculated by using normals of a curved surface approximated to cubic surfaces.

Lane et al. (1980) and Clark (1979) rendered curved surfaces after subdividing them into small polygons. That is, curved surfaces are subdivided into subpatches until flat enough, then the subpatches are regarded as polygons, and finally a polygon-oriented scanline algorithm is employed. In Clark's method, curved surfaces are subdivided into subpatches before scan-conversion, while in Lane-Carpenter's method curved surfaces are dynamically subdivided for every scanline. Lane-Carpenter's method has the disadvantage that gaps arise between approximated polygons due to the difference between the approximated polygons and the original surface patches. In the subdivision methods taking into account surface flatness, only when the tolerance of surface flatness is within one pixel size do the silhouettes become smooth.

Griffiths (1984) dealt with curved surfaces in the parametric space. In this method, a curved surface is decomposed into grid cells; some of them intersecting with a scanline are further divided. Then, linear interpolation is employed to obtain their depths and intensities. In this method, silhouettes are extracted by using normal vectors stored in the grid cells. Pueyo and Brunet (1987) improved upon Griffiths' method; they proposed that curved surfaces are decomposed into grid cells beforehand, intersections of the restricted scanline and curved surfaces are calculated without interpolation (making use of the y-coordinates stored in the grid cells), and interpolation in parametric space is employed for intersections of the other scanlines. Silhouettes are detected by using the z-components of normal vectors stored in the grid points. One of the disadvantages of the previous two methods is missing those silhouettes formed by a smaller loop than the grid span.

3 Outline of the algorithm

Rational Bézier surfaces are used in this paper, because almost all surfaces, such as B-splines and NURBS, can be converted into rational Bézier surfaces. In this paper, curved surfaces are subdivided into subpatches on a scanline. These subpatches are processed as polygons with curved edges, and intersections of every subpatch and the scanline are calculated. Finally, traditional polygon-oriented scanline algorithms represented by Watkins's algorithm (1970) are employed to display the images.

Outline of the proposed algorithm is as follows:

Step 1: Calculate a bounding box on the projection plane for each surface patch after the perspective transformation of every control point of each surface patch.

Step 2: Find surface patches possibly intersecting with the scanline, and subdivide them into subpatches on the scanline.

Step 3: Calculate every span (called a scan segment) where subpatches intersect with the scanline.

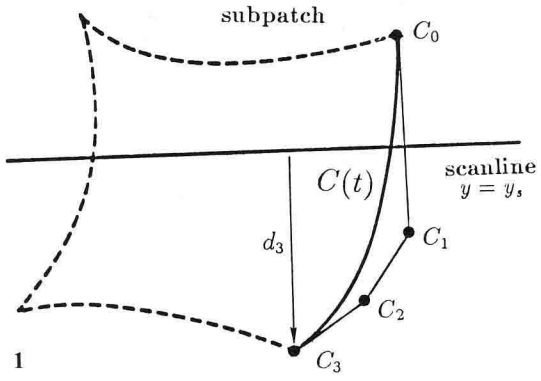
Step 4: Find visible parts on each scan segment.

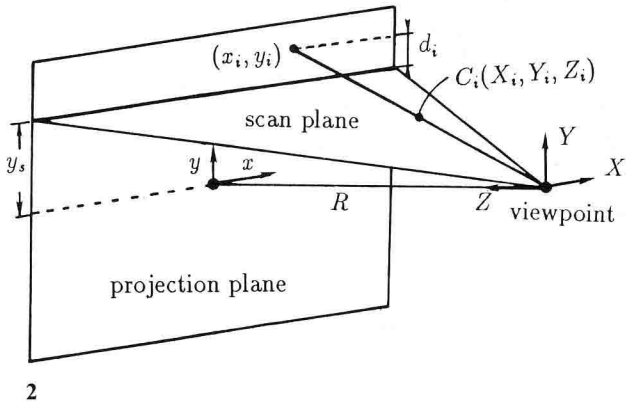
Step 5: Calculate shadow sections on each visible part.

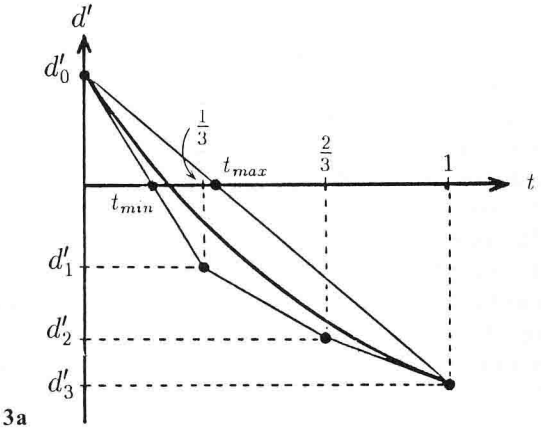
Step 6: Display intensity of each pixel by using a smooth-shading algorithm.

In Step 1, a convex hull property of each Bézier surface is useful to determine its bounding box. That is, the bounding box is determined by using the minimum and maximum values of the coordinates (x, y) of control points. In Step 2, each subpatch is recursively subdivided until a desired degree of surface flatness is achieved. To find intervals of u, v parameters that overlap with each scanline, Bézier clipping is employed. In Step 3, the spans of scan segments are calculated by using intersections of a curved boundary of each subpatch and the scanline. For trimmed patches, the scan segments are clipped by the trimming curves by using Bézier clipping in parametric space.

The proposed method basically belongs to polygon-oriented scanline algorithms, but Steps 2, 3,







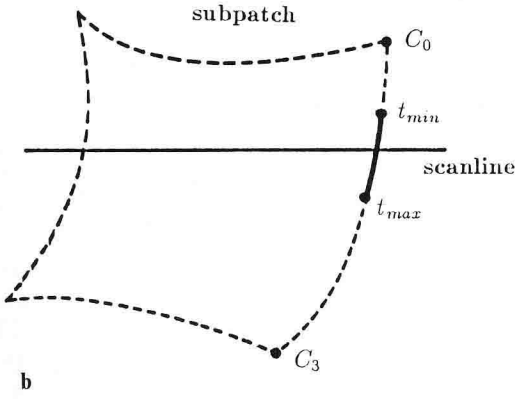


Fig. 1. Intersection test of a boundary curve and a scanline

Fig. 2. Relationship between control points of a curved surface and a scan plane

Fig. 3a, b. Calculation of intersections in parametric space and clipping a boundary curve

and 5 are different from them. In the following sections, these steps are discussed.

4 Intersections of curves and a scanline

In the proposed method, after subdividing curved surfaces into subpatches on each scanline, subpatches are scan-converted into scan segments. For the convenience of explaining this method, first the calculation method of the intersections between curves and the scanline in Step 3 is described, although it follows Step 2.

Let's assume that a curved surface is defined by rational Bézier surface of degree n [refer to Eq. (4)], and the boundary curves of the surface $P(u, v)$ are expressed by $P(u, 0)$, $P(u, 1)$, $P(0, v)$, and $P(1, v)$. These curves are also expressed by rational Bézier curves of degree n. Then, each boundary curve $C(t)$ in a space is defined by the following.

$$C(t) = \frac{\sum_{i=0}^{n} W_i C_i B_i^n(t)}{\sum_{i=0}^{n} W_i B_i^n(t)}, \tag{1}$$

where $C_i(X_i, Y_i, Z_i)$ $(i = 0, 1, \ldots, n)$ are control points, W_i are its weights, and B is the Bernstein basis polynomial expressed by $B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$.

The calculation for the intersections of a projected curve $C(t)$ and a scanline $y = y_s$ (Fig. 1) is discussed in the following. Let's assume that control points $C_i(i = 0, 1, \ldots, n)$ are defined in the eye coordinate

system (X, Y, Z), as shown in Fig. 2; the origin is set to the viewpoint and the z axis is perpendicular to the projection plane. After perspective transformation of C_i, the curve $C(t)$ is expressed by the following rational Bézier function (small letters are used to express perspective-transformed coordinates):

$$x(t)=\sum_{i=0}^{n} W_i X_i B_i^n(t)/z(t),$$

$$y(t)=\sum_{i=0}^{n} W_i Y_i B_i^n(t)/z(t), \text{ and}$$

$$z(t)=\sum_{i=0}^{n} W_i Z_i B_i^n(t), \tag{2}$$

where $z(t)$ corresponds to the depth of the curve. The homogenous coordinates (x_i, y_i, w_i) of each control point of the projected curve are given by $x_i=X_i/Z_i$, $y_i=Y_i/Z_i$, $w_i=RW_i/Z_i$, where R is the distance from the viewpoint to the projection plane. Then the curve on the projection plane is expressed by

$$x(t)=\sum_{i=0}^{n} w_i x_i B_i^n(t)/\sum_{i=0}^{n} w_i B_i^n(t) \text{ and}$$

$$y(t)=\sum_{i=0}^{n} w_i y_i B_i^n(t)/\sum_{i=0}^{n} w_i B_i^n(t). \tag{3}$$

Because the line equation of the scanline is expressed by $y-y_s=0$ (the scan plane is expressed by $Y-(y_s/R)Z=0$ in 3D space), the following equation is derived by substituting this into Eq. (3):

$$\sum_{i=0}^{n} d_i' B_i^n(t)=0, \tag{4}$$

where

$$d_i'=w_i(y_i-y_s)=w_i d_i.$$

d_i corresponds to the distance between the scanline and a control point C_i on the projection plane ($d_i'=(Y_i-(y_s/R)Z_i)W_i$ in 3D space, as shown in Fig. 2). If a parameter t, which satisfies Eq. (4) is calculated, then the depth z and x coordinates of the intersection on the projection plane can be derived.

The parameter t in Eq. (4) can be solved by using the Newton-Raphson method, because the equation is a polynomial of degree n; however, the calculation is not always robust. To overcome the problem, the authors employ Bézier clipping method (Nishita et al. 1990; Sederberg and Nishita 1990). When the curve intersects with the scanline, the root of Eq. (4) always exists between the intersections of the convex hull determined by the vertexes $(i/n, d_i')$ $(i=0, 1, \ldots, n)$ and the t-axis, because the curve defined by Eq. (4) is a non-parametric function (Fig. 3a), while the curve does not intersect with the scanline when $d_i'(i=0, 1, \ldots, n)$ are positive for all i or negative for all i. Only when d_i' has both positive and negative value (in this condition the roots exist), the outside of the interval $[t_{\min}, t_{\max}]$ of the curve is clipped away, as shown in Fig. 3a. As the intersections of the convex hull formed by the control points of the clipped curve (as shown in Fig. 3b) and the t-axis are recursively calculated, the interval where the roots exist becomes narrow. The root of t converges with the advance of the clipping process. If Eq. (4) has more than two roots, the ratio of the convergence of the interval becomes small. In this case, the curve is divided into two curves and the same process as mentioned above is continued for each divided curve; thereafter all of the roots can be calculated. This clipping method consists of two processes, that is, calculating the interval of t and clipping a curve. Generally speaking, the latter process needs extensive calculation time, because curves in a space have the three components x, y, and z, while in our method, calculation time can be saved, because only one component d' is used to clip a curve; de Casteljau's subdivision algorithm is employed for the clipping.

5 Clipping a curved surface on a scanline

The method generating for subpatches intersecting with the scanline is described here. In Lane-Carpenter's method, a curved surface is subdivided into four subpatches. The subpatches on the scanline are extracted, and the subdivision is recursively continued until surface flatness is satisfied. In this method, not only the subdivision process, but also the extraction of the subpatches existing on the scanline are necessary. We propose a more efficient method of generating subpatches on the scanline. A region of a curved surface is recursively subdivided by clipping away where the surface does not intersect with the scanline. In many cases, a sub-

patch intersects with several scanlines. In this paper, subpatches intersecting with every few scanlines (e.g., every three or four scanlines) are generated, and the intersections of the boundary curves of subpatches and the scanline are calculated.

Assuming that the coordinates of a control point are (X_{ij}, Y_{ij}, Z_{ij}) with weights W_{ij} in the eye coordinate system, rational Bézier surfaces of degree n are defined as

$$X(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{n} W_{ij}\,X_{ij}\,B_i^n(u)\,B_j^n(v)}{\sum_{i=0}^{n}\sum_{j=0}^{n} W_{ij}\,B_i^n(u)\,B_j^n(v)},$$

$$Y(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{n} W_{ij}\,Y_{ij}\,B_i^n(u)\,B_j^n(v)}{\sum_{i=0}^{n}\sum_{j=0}^{n} W_{ij}\,B_i^n(u)\,B_j^n(v)}, \text{ and}$$

$$Z(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{n} W_{ij}\,Z_{ij}\,B_i^n(u)\,B_j^n(v)}{\sum_{i=0}^{n}\sum_{j=0}^{n} W_{ij}\,B_i^n(u)\,B_j^n(v)}. \tag{5}$$

When the range for generating subpatches is a band between the two scanlines $y_s^{(1)}$ and $y_s^{(2)}$ $(y_s^{(1)} > y_s^{(2)})$, as shown in Fig. 4, equations of the scan planes determined by each scanline and the viewpoint are

$$Y-(y_s^{(k)}/R)\,Z=0 \qquad (k=1,2). \tag{6}$$

If the functions $D^{(1)}$ and $D^{(2)}$ are expressed as

$$D^{(k)}(Y,Z)=Y-(y_s^{(k)}/R)\,Z \qquad (k=1,2),$$

the region formed by these two scanlines satisfies the condition $D^{(1)}\le 0$ and $D^{(2)}\ge 0$. From this, the interval of u and v intersecting with the scan planes defined by Eq. (6) satisfies the conditions

$$D^{(1)}=\sum_{i=0}^{n}\sum_{j=0}^{n}(Y_{ij}-(y_s^{(1)}/R)\,Z_{ij})\cdot W_{ij}\,B_i^n(u)\,B_j^n(v)\le 0 \quad \text{and}$$

$$D^{(2)}=\sum_{i=0}^{n}\sum_{j=0}^{n}(Y_{ij}-(y_s^{(2)}/R)\,Z_{ij})\cdot W_{ij}\,B_i^n(u)\,B_j^n(v)\ge 0. \tag{7}$$

That is,

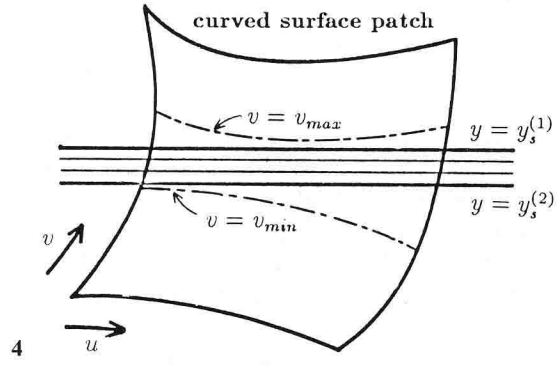
$$\sum_{i=0}^{n}\sum_{j=0}^{n} D_{ij}^{(1)}\,B_i^n(u)\,B_j^n(v)\le 0,$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} D_{ij}^{(2)}\,B_i^n(u)\,B_j^n(v)\ge 0, \quad \text{and}$$

$$D_{ij}^{(k)}=(Y_{ij}-(y_s^{(k)}/R)\,Z_{ij})\,W_{ij} \qquad (k=1,2). \tag{8}$$

Note that $D_{ij}^{(k)}=(y_{ij}-y_s^{(k)})\,w_{ij}$ on the projection plane. The Bézier clipping described in the previous section is also available to calculate the interval of u and v in Eq. (8).

A curved surface is clipped taking into account the obtained interval of u and v. If flatness of the clipped surface, a subpatch, is not enough, the subpatch is divided into two subpatches and this pro-



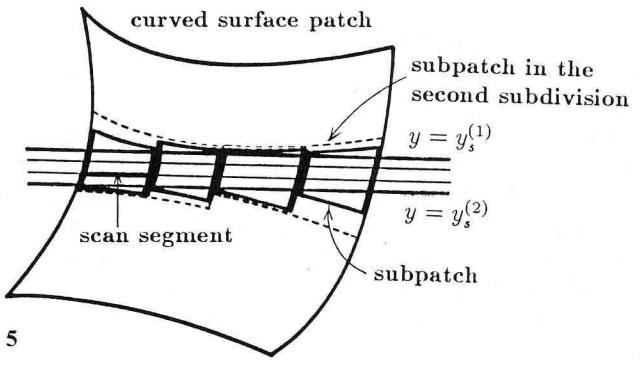


Fig. 4. Intersection of a group of scanlines and a surface patch

Fig. 5. Subpatches on a scanline

cess is continued until every subpatch satisfies the flatness tolerance given in advance. Finally, several subpatches intersecting with the scanline are generated, as shown in Fig. 5. Surface flatness is measured by the maximum distance between the curved surface and the plane determined by three corner control points (Whitted 1978; Lane et al. 1980).

6 Hidden surface removal and shading

After generating subpatches, a line segment between intersections of a subpatch and each scanline located between $y=y_s^{(1)}$ and $y=y_s^{(2)}$ is calculated by using the method described in the previous section. This line segment is called a scan segment. Scan segments can be reckoned as straight lines, because subdivided subpatches are flat enough. Except for the original boundaries, boundary curves consisting of subpatches are classified into two types. One is a boundary caused by clipping away the outside of the region determined by the interval of u or v obtained by Eq. (8), and the other is a boundary caused by dividing a subpatch into two subpatches. In the calculation of intersections of subpatches and the scanline, it is enough to examine only the latter's boundaries (usually, two edges as shown in bold lines in Fig. 5), because the former never intersect with the scanline.

After calculating scan segments, visible parts of each scan segment are determined by using a traditional scanline algorithm, and Phong's shading algorithm (1975) is employed to display curved surfaces. In this paper, penetrations of each curved surface are allowed; hidden surfaces are removed by taking account of the intersection of scan segments in the depth direction.

For the visibility test and smooth shading mentioned above, a depth z and a normal vector at both endpoints of a scan segment are required. Normal vectors at both endpoints of a scan segment are only slightly different, because of flatness of the subpatch. When the difference between the normal vectors at both endpoints is greater than a threshold, the subpatch is divided again. Therefore, accurate intensities can be obtained, even if a linear interpolation is employed to calculate the normal vectors on the scan segment.

To discuss accuracy of shading proposed here, some previous methods are referred to here. In Schweitzer and Cobb's method (1982), a cubic interpolation is employed to calculate normal vectors on a scan segment. However, the accuracy of the normal vectors is not sufficient for faithful display of an original curved surface, because the normal vectors on the scan segment are interpolated by using only the normal vectors at the intersections of the scanline and a fairly large subpatch (and/or a silhouette). In Pueyo and Brunet's method (1987), a curved surface is decomposed (equidistantly in parametric space) into grid cells, and linear interpolation is employed for the normal vectors on the grid cells. As the difference between the normal vectors at the adjacent grid points depends upon the size of grid cells, the grid cells should be small enough to obtain accurate normal vectors. In order to obtain more precise coordinates and normals at pixels within the scan segments, a marching technique, such as Satterfield and Roger's method (1985), may be effective, even though their method is developed for generating contour lines from a B-spline after triangular mesh approximation.

7 Silhouettes of a curved surface

The detection of accurate silhouettes of a curved surface is one of the important elements for displaying realistic curved surfaces. As described in Sect. 2, traditional calculation methods are neither always robust nor so accurate for silhouette edges (Blinn 1978; Whitted 1978; Schweitzer and Cobb 1982). Even some methods addressing these problems still have some disadvantages, such as a very complicated process and missing silhouettes formed by relatively small loops. In the subdivision methods taking into account surface flatness (Lane et al. 1980; Clark 1979), only when the tolerance of surface flatness is within one pixel do the silhouettes become smooth; in this case every curved surface needs to be excessively subdivided even though it does not always have silhouettes. To solve these problems, the authors propose the following efficient subdivision method.

It is useful to classify all curved surfaces into two types – those which probably have silhouettes, and those which never have them before scan-conversion, because all the curved surfaces do not necessarily have silhouettes. After subdividing curved surfaces into subpatches on the scanline, only the subpatches probably having the silhouettes are examined to see whether they really do have silhou-

ettes or not. If they do, the subdivision process is applied until the flatness is satisfied. (The lower tolerance of flatness should be set to the subpatches with silhouette). This method never leaves any silhouette undetected. In the following paragraph, we will discuss how to examine whether a curved surface has a silhouette or not.

First, for closed surfaces the classification into a front face, a back face, or the other (i.e., surfaces probably having silhouettes) is necessary. A front face has a normal vector toward the viewpoint throughout the surface (an inner product of the viewing vector and the normal vector is positive), while a back face has a normal vector away from the viewpoint. The front/back test for curved surfaces is described in the Appendix.

Curved surfaces requiring scan-conversion can be reduced in number by culling back faces in the step of the wedge test, because of the invisibility of back faces. (Of the total CPU time, 15% was cut down by culling back faces in our experiment.)

A bisectional method is employed to calculate the intersection of silhouettes and each scanline, because the interval between the parameters at the endpoints of a scan segment is very small. That is, if normal vectors N_1 and N_2 at the endpoints (Q_1 and Q_2 in Fig. 6) of a segment have an opposite direction with respect to the direction of the viewpoint, subdivision of the subpatch is executed, because the scanline intersects with the silhouette (at Q_3 in Fig. 6). The subdivision process is continued until the viewpoint direction component of both normal vectors at the endpoints becomes smaller than a specified tolerance.

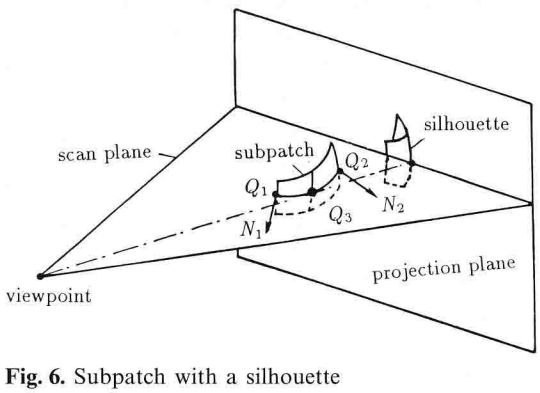


Fig. 6. Subpatch with a silhouette

8 Trimmed surfaces

The intersection test between a scan segment and trimming curves (in Step 3 in Sect. 3) is performed in parametric space, and the scan segments are clipped by the trimming curves. The trimming curves representing holes on the surface are expressed by Bézier form with degree n in parametric space and expressed by

$$u(t)=\sum_{i=0}^{n} u_i B_i^n(t) \quad \text{and}$$

$$v(t)=\sum_{i=0}^{n} v_i B_i^n(t), \tag{9}$$

where $(u_i, v_i)(i=0,\ldots,n)$ are control points of a trimming curve. For simple calculation, translation and rotation are performed; one endpoint of the scan segment is set as the origin and the scan segment coincides with the u-axis (Fig. 7). Let's denote the coordinate system as (u', v') after transformation. Because the scan segments lie on the line $v'=0$, the following equation is obtained by substituting curve Eq. (9) to the line equation:

$$v'(t)=\sum_{i=0}^{n} v_i' B_i^n(t). \tag{10}$$

The above equation is a nonparametric Bézier function whose control points are $(i/n, v_i')$. This equation is also solved by Bézier clipping. After solving parameter t satisfying the above equation, the intersection point on the scan segment is obtained from t. Even in the case where there is no intersection on the scan segment, sometimes the scan segment should be removed (see $Q_2 Q_3$ in Fig. 7b); whether the scan segment should be removed or not is determined by counting the number of positive intersections on the u' axis. That is, if the number of intersections is even, the scan segment is surrounded by the trimming curves.

9 Shadowing

Even though the papers referred to here use scanline algorithms (Blinn 1978; Whitted 1978; Clark 1979; Lane et al. 1980; Schweitzer and Cobb 1982; Griffiths 1984; Pueyo and Brunet 1987) none of them mentions shadowing. Shadows are one of the important elements for displaying realistic images.

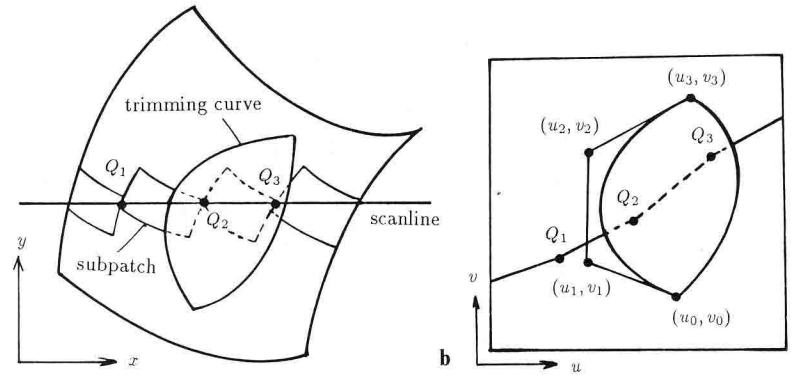


Fig. 7a, b. Intersection test between a scan segment and trimming curves. **a** Projection plane, **b** parametric plane

A calculation method for shadowed sections on each scan segment is discussed here. Shadows are invisible sections when a light source is reckoned as a viewpoint. Therefore, a shadow cast by a curved surface can be calculated as follows. If a triangle formed by both endpoints of a scan segment and the light source (Fig. 8) intersects with a curved surface, the shadows due to the curved surface are cast on at least a part of the scan segment. By reckoning the light source as a viewpoint, the triangle formed by the scan segment and the light source is treated as a scan plane; the scanline algorithm mentioned before can be employed. That is, the shadow sections on the scan segment can be determined by using the intersection test between the scanline and curved surfaces when viewed from the light source.

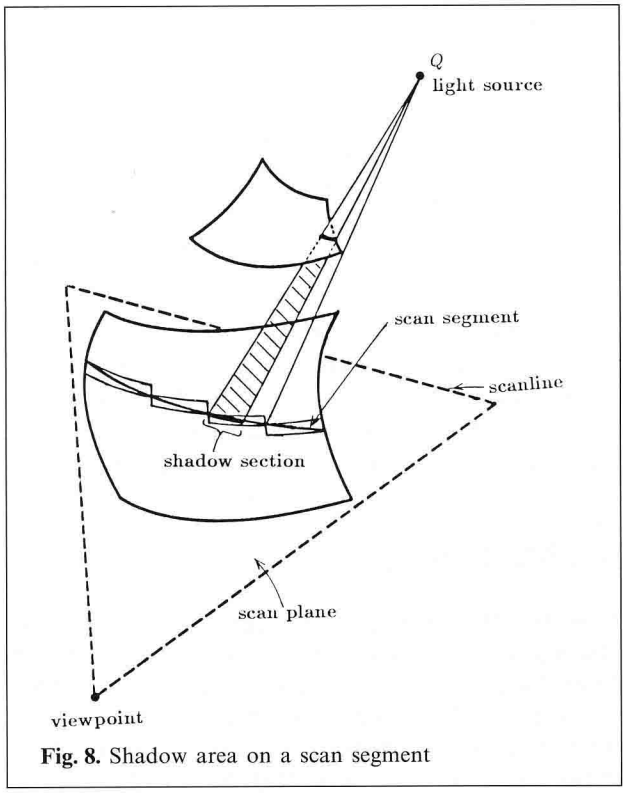


Fig. 8. Shadow area on a scan segment

10 Examples

Four examples are shown in Fig. 9. Figure 9a shows a teapot as an example of standard data. It consists of 32 patches, and the CPU time was 27.3 s (144.6 s in the case with shadows) when IRIS-4D/120GTX was used with a screen 500×500. Table 1 shows relationships among the tolerance of the parameter t regarding the calculation of intersections of a scanline and subpatches, the average number of iterations, and the CPU time. It is evident that iterations are fairly few, because of quick convergence due to the fact that the shapes of boundaries of every subdivided subpatch are close to straight segments. When the tolerance is set at 10^{-3}, even one patch covering all of the screen size (1000×1000) is displayed with accuracy within one pixel. Therefore, it is set at 10^{-3} in the examples. Even when the resolution of the screen becomes higher and the tolerance is set at 10^{-7}, the average number of iterations increases by only 0.8, and the CPU time increases by less than 1%. Calculation time for generating subpatches takes 60% of all calculation time, and for calculating intersections of subpatches and a scanline and hidden surface removal, 22%.

Figure 9b shows an example of texture mapping to Fig. 9a. Figure 9c shows an example of a

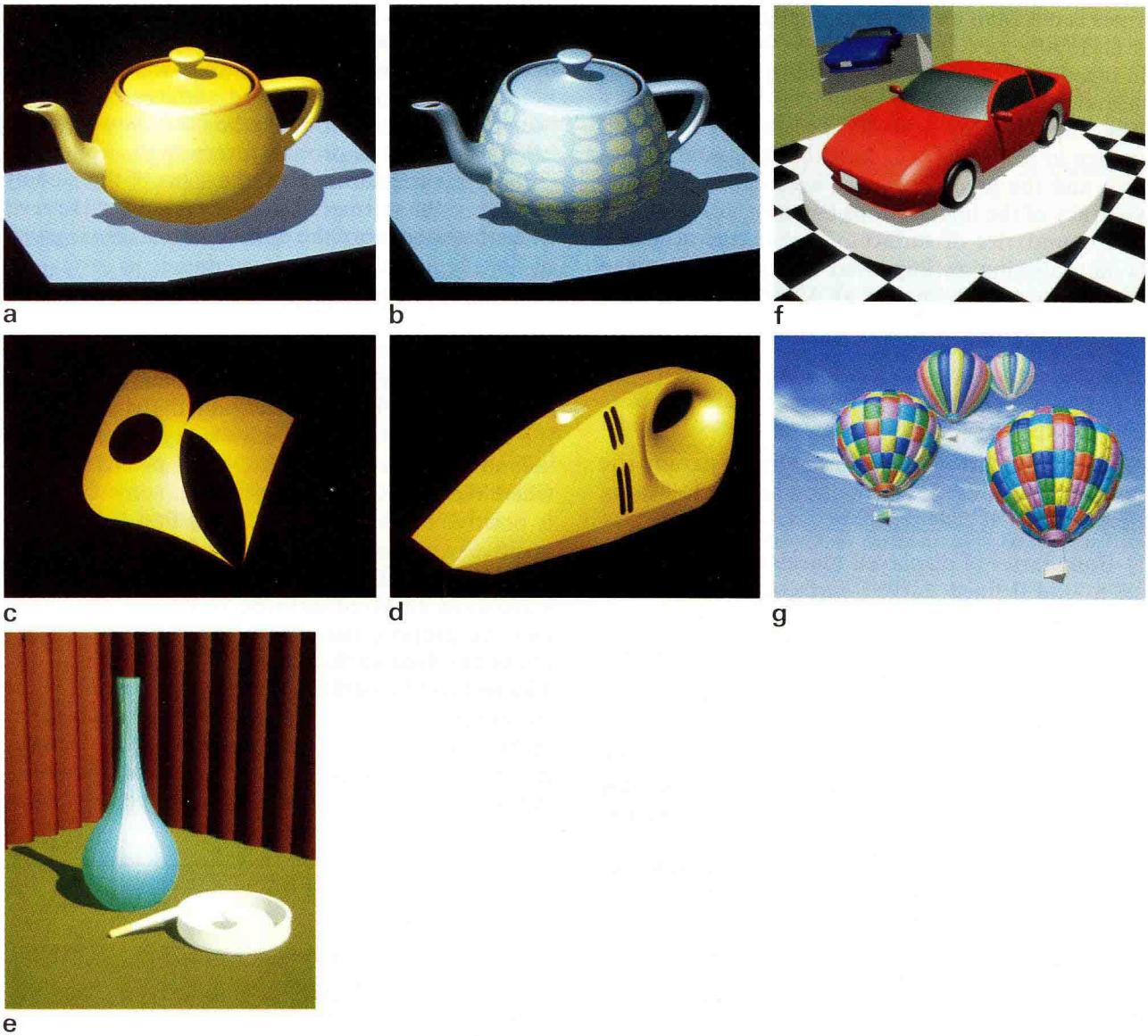


Fig. 9a–g. Examples. **a** teapot, **b** textured teapot, **c** trimmed patch, **d** cleaner, **e** base (133 patches), **f** automobile (555 patches), **g** balloons (1064 patches)

Table 1. Relationship between the tolerance, average iterations, and CPU time (in the case of Fig. 9a)

Tolerance	Average iterations	CPU time (s)
10^{-3}	1.6	27.3
10^{-5}	2.1	28.9
10^{-7}	2.4	29.1

trimmed patch. Figure 9d shows a commercial product design (a hand-held vacuum cleaner) consisting of 56 patches; the CPU time is 45.6 s. Figure 9e shows a base and an ash tray consisting of 133 patches. Figure 9f shows an automobile composed of 555 Bézier patches; the CPU time is 55.3 s (222.0 s in the case with shadows). In Fig. 9g, depicting hot-air balloons considered foggy effect, the CPU time is only 228.8 s, although all of the balloons consist of 1064 patches. These examples show that the greater the number of patches, the more effective the method is. Even though raytracing algorithms are usually very powerful for obtaining realistic images, for hidden surface removal (excluding shading) the method proposed here is five

times faster than our raytracing algorithm (Nishita et al. 1990).
In these examples, a multi-scanning method (Nishita and Nakamae 1984) developed by the authors was employed for anti-aliasing, the area of each surface in a pixel is calculated by trapezoidal integral, and the precision of the area depends on the accuracy of the intersections between sub-scanlines and boundaries of surfaces. That is, the accuracy of anti-aliasing is improved. Anti-aliased image composition (Nakamae et al. 1986) were used for Fig. 9g.

11 Conclusions

A robust and fast method for rendering parametric surfaces is proposed. The method has the following advantages:

1) Boundaries of displayed objects are quite faithful to the defined curved surfaces, and therefore extremely smooth, because intersections of subpatches and each scanline are calculated accurately and robustly. No gap arises between subpatches. Intersections can be calculated by few iterations (about two iterations).
2) Even in the area where the changing of normal vectors is drastic, the intensity is accurate, because curved surfaces are sampled in proportion to curvature of the surfaces.
3) Shadowed boundaries are always smooth, because shadows faithful to the defined curved surfaces can be displayed.
4) In terms of calculation time, it is low cost especially when a number of patches exist in a scene.
5) The accuracy of anti-aliasing is improved, because of calculating the accurate intersections.

Acknowledgements. We would like to thank Dr. Thomas W. Sederberg for his discussion about the method for calculating intersections by Bézier clipping when the first author stayed at Brigham Young University.

Appendix. Front/back test for curved surfaces

The normal vector at a point $(x(u, v), y(u, v), z(u, v))$ on surface $P(u, v)$ is defined by a vector product of $P_u(u, v)$ and $P_v(u, v)$ (i.e., $P_u = \partial P(u, v)/\partial u$, $P_v = \partial P(u, v)/\partial v$). If the condition $P(u, v) \cdot P_u(u, v) \times P_v(u, v) < 0$ is satisfied for all $u(0 \leqq u \leqq 1)$ and $v(0 \leq v \leq 1)$ in the eye coordinate systems, then the curved surface is a front surface toward the viewpoint. To find out the curved surfaces having silhouettes, this condition may be used; however, the calculation cost is probably very high. Following perspective transformation, the signs of the z-component of the normal vectors are used for the test. The z-component of the normal vector, N_z, is given by

$$N_z(u, v) = \frac{\partial x(u, v)}{\partial u}\frac{\partial y(u, v)}{\partial v} - \frac{\partial y(u, v)}{\partial u}\frac{\partial x(u, v)}{\partial v}. \quad (11)$$

For B-spline surfaces, Elber and Cohen (1990) used the fact that if the first term on the right side of the above equation is positive everywhere and the second negative everywhere, N_z is positive everywhere (i.e., front face). However, in some cases, N_z is positive everywhere even if the first term is not positive; e.g., x or y component of the partial derivative includes 0. In such cases, their method may leave some unsolved surfaces. To address this problem, we propose the following method, which reduces unsolved surfaces.
The test can be performed on the projection plane by using only the x and y components, because in Eq. (11) the sign of the z-component of the normal vector is defined only by the x and y components of P'_u and P'_v, where P' is a perspective trans-

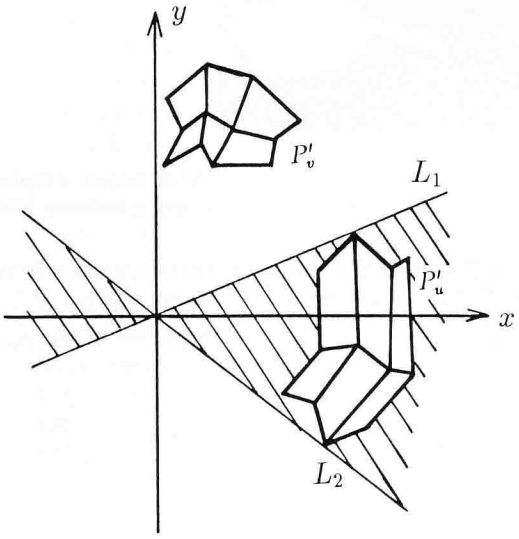


Fig. 10. Hodograph for calculating directions of normals

formed curved surface. P'_u and P'_v are rational Bézier surfaces and the range of their tangent direction is bound by the hodograph of non-rational surfaces (Sederberg and Meyers 1988). Derivatives of non-rational Bézier surfaces can be obtained from the control points geometrically. For example, the control points P'_{uij} of P'_u are expressed by $n(P'_{i+1,j}-P'_{i,j})$ $(i=0, 1, \ldots, n-1, j=0, 1, \ldots, n)$. For a degenerate Bézier surface with a side collapsed to a point, the derivative at the point cannot be obtained. In this case, it can be obtained by approximated control points, which are very close to the original point.

The range of their tangent direction of P'_u and P'_v can be obtained by hodograph (Sederberg and Meyers 1988); the direction of the tangent with respect to the parameter u exists in the wedge area intercepted by the lines L_1 and L_2, which hold P'_u, as shown in Fig. 10 (hatching part). The equations of the lines L_1 and L_2 are defined by

$$f_1(x, y)=a_1 x+b_1 y=0 \quad \text{and}$$
$$f_2(x, y)=a_2 x+b_2 y=0. \tag{12}$$

Every control point P'_{uij} belonging to P'_u locates in the positive side of the line L_1, while P'_{uij} in the negative side of the line L_2. That is, the straight lines satisfy the conditions

$$\forall P_{uij}: f_1(x_{uij}, y_{uij}) \geq 0$$
$$\forall P_{uij}: f_2(x_{uij}, y_{uij}) \leq 0. \tag{13}$$

Therefore,

(a) If $\forall P_{vij}$: $f_1(x_{vij}, y_{vij}) \geq 0$ and $f_2(x_{vij}, y_{vij}) \geq 0$ then a front face.

(b) If $\forall P_{vij}$: $f_1(x_{vij}, y_{vij}) \leq 0$ and $f_2(x_{vij}, y_{vij}) \leq 0$ then a back face.

(c) Other than the above, a twisted face. (14)

In other words, if any P_v overlaps with the wedge intercepted by L_1 and L_2, then the curved surface is a twisted surface. Note that the curved surface classified into a twisted surface does not always have a silhouette. For a more detailed test, the curved surface must be further subdivided, but it is enough to find out only the probability of having silhouettes. In this process of face classification, if L_1 and L_2 cannot be obtained the surface probably has silhouettes.

Note that in Elber and Cohen's method (1990), the case of Fig. 10 is classified into a twisted face $\left(\text{i.e., } \frac{\partial y(u, v)}{\partial u} \text{ includes } 0\right)$ although it is classified into a front face with our method.

References

Blinn JF (1978) Simulation of wrinkled surfaces. Comput Graph 12(3):286–292

Clark JH (1979) A first scan-line algorithm for rendering parametric surfaces. Comput Graph 13(2):174

Elber G, Cohen E (1990) Hidden curve removal for free form surfaces. Comput Graph 24(4):95–104

Griffiths JG (1984) A depth-coherence scanline algorithm for displaying curved surfaces. CAD 16(2):91–101

Lane JM, Carpenter LC, Whitted T, Blinn JF (1980) Scan line methods for displaying parametrically defined surfaces. Commun ACM 23(1):23–34

Nakamae E, Harada K, Ishizaki T, Nishita T (1986) A montage: the overlaying of the computer generated images onto a background photograph. Comput Graph 20(4):207–214

Nishita T, Nakamae E (1984) Half-tone representation of 3-D objects with smooth edges by using a multi-scanning method. Trans IPSJ 25(5):703–711

Nishita T, Sederberg TW, Kakimoto M (1990) Ray tracing rational trimmed surface patches. Comput Graph 24(4):337–345

Phong BT (1975) Illumination for computer generated pictures. Commun ACM 18(6):311

Pueyo X, Brunet P (1987) A parametric-space-based scan-line algorithm for rendering bicubic surfaces. IEEE 7(8):17–24

Satterfield SG, Rogers DF (1985) A procedure for generating contour lines from a B-spline surface. IEEE CGA 5(4):71–75

Schweitzer D, Cobb ES (1982) Scanline rendering of parametric surfaces. Comput Graph 16(3):265–271

Sederberg TW, Meyers RJ (1988) Loop detection in surface patch intersections. CAGD 5(2):161–171

Sederberg TW, Nishita T (1990) Curve intersection using Bézier clipping. CAD 22(9):538–549

Watkins GS (1970) A real-time visible surface algorithm. University of Utah Computer Science Department UTEC-CSC-70-101, NTIS AD-762 004

Whitted T (1978) A scan line algorithm for computer display of curved surfaces. Comput Graph 12(3):26

For author's biographies and photos see pp. 267 and 268.