A Parametric-Space-Based Scan-Line Algorithm for Rendering Bicubic Surfaces

X. Pueyo and P. Brunet
Universitat Politècnica de Catalunya

This article presents a new scan-line algorithm for displaying bicubic surfaces. Patches are decomposed on regions of constant sign of the z component of the normal before the scan process. Most of the computations are done in parametric space. The algorithm computes the intersection of the surfaces with only a restricted subset of scan planes and obtains the intersection with other scan planes by linear interpolation between exact intersections. A bound of the algorithm's error is given. Finally, the new method is compared with Whitted's algorithm.

In recent years hidden-surface-removal algorithms have been proposed to render curved surfaces, especially bicubic (or bipolynomial) patches. Most of these algorithms use the scan-line principle to determine the scene visibility. Lane, Carpenter, Whitted, and Blinn[1] proposed three different methods. The first algorithm is a subdivision method that stops when a desired degree of surface flatness is achieved. Then subpatches are approximated by polygons processed by a polygon-oriented algorithm. Whitted's method computes the exact intersection of some curves in the patch with every scan plane and approximates the patch's intersection with the plane by a polygonal line defined by these points. The exact intersection curves are the cubics defining the boundaries of the patch, isoparametric curves in the patch, and cubic approximations of the silhouettes. Finally, Blinn proposes a method that is essentially a z-buffer algorithm in which numerical problems are simplified using the intersection of the scan planes with the boundaries and the silhouettes of the patch.

Lane and Carpenter[2] later improved and generalized their algorithm, and Schweitzer and Cobb[3] proposed an algorithm similar to Whitted's method that offered higher quality silhouettes.

Some methods compute curved-surface visibility in parametric space. These algorithms use different techniques employed in polygon-oriented methods. Strasser[4] and Forest[5] use a z-buffer algorithm; Griffiths'[6,7] and Ohno's[8] algorithms are based on depth comparison between points of a grid defined in the patch and some subpatches; Hornung et al.[9] simply compute the boundaries of constant visibility areas in the patch.

Griffiths[10] has proposed a scan-line algorithm that does some work in the parametric plane. This algorithm obtains an approximation of the patch silhouettes in the parametric space. Griffiths' method approximates the intersection of the scan plane and the patch with chains of straight segments instead of considering the segments separately, as other algorithms do. In this way his method speeds up the visibility calculation considerably.

We will present a new scan-line algorithm for curved surfaces that does most of the computations in the para-

0272-1716/87/1100-0017$01.00 © 1987 IEEE

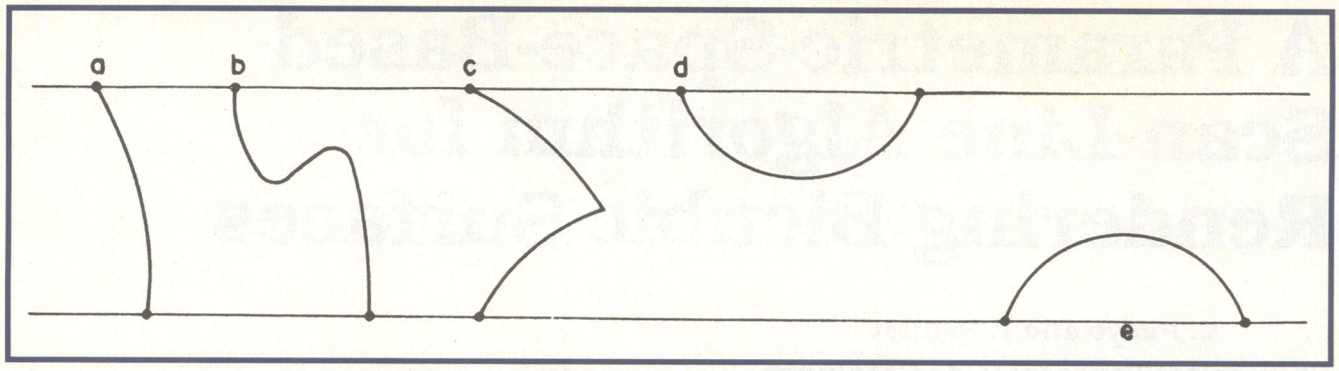

Figure 1. Possible types of curves in a band defined by two scan planes.

metric space and computes the intersection of the surfaces with only a restricted set of scan planes. Then we will analyze the errors produced by the proposed algorithm. Finally, we present the performance characteristics of this method.

The algorithm

The work in the parametric plane is performed to a specified precision *m* used to define an mxm grid in the patch. For every point of this grid, we compute the exact values of *y* (vertical coordinate) and the z component of the normal. Other values of *y* and the z component of the normal are obtained by linear interpolation along parametric grid lines.

Before we start the scan process, every patch is decomposed in regions of constant normal sign, that is, without interior silhouettes. During the computation of these regions we find the singular points. These are local and global minima and maxima with respect to the *y* coordinate, and the vertices of the regions. These will be used in the scan process.

One of the main goals of the new method is to compute the intersection of the surfaces with only a restricted subset of scan planes (*exact intersection planes*) and to obtain the intersection with other planes by interpolation between pairs of consecutive exact intersection planes. This interpolation will be correct if there are no qualitative changes of the scene between the two consecutive exact intersection planes. This happens when

1. The number of segments (portion of intersection connecting two consecutive edges of the region) of a region is constant for all horizontal planes in the band defined by the exact intersection planes.
2. The number of active regions (regions intersected by the current scan plane) is constant in the band.
3. Pairs of points to be interpolated of the exact intersection planes are connected by curves of continuous slope.

We now present the different cases of qualitative scene change. The scan planes containing these changes will be called *singular planes*. Let's take an arbitrary band of scan planes and a point in the highest plane. A curve (patch boundary or silhouette) leaving this point may or may not arrive at the lowest scan plane. If it does arrive, three different cases may be found:

1. The curve has a continuous slope and has neither minima nor maxima in the band (Figure 1, a).
2. The curve's slope is continuous and has a minimum and a maximum in the band (Figure 1, b). The number of segments changes in the planes containing the minimum and maximum.
3. The curve's slope is not continuous (Figure 1, c); it contains a vertex.

If the curve does not arrive at the lowest plane in the band, it contains a minimum or a vertex (Figure 1, d), and the number of segments changes.

Now let's take a point in the lowest scan plane of the band. A curve leaving this point and arriving at the highest plane is of type a, b, or c of Figure 1. If the curve does not reach the highest plane, it contains a maximum or a vertex (Figure 1, e), and the number of segments changes.

So singular planes are those containing a minimum, a maximum, or a vertex. A set of consecutive planes with no qualitative changes of the scene is known as an *admissible band*. A working band will be an admissible band with a maximum of *n* scan planes, where *n* is a new parameter given by the user. For a given exact intersection plane we have two bands. One is defined by the exact intersection plane and the *n*th scan plane after it. The other is defined by the exact intersection plane and the next plane containing a singular point. The narrower of these bands is taken as the *working band*.

Patch decomposition

To decompose the surface patches in regions where the sign of the z component of the normal is constant, we need to compute the silhouettes. These are obtained in a manner similar to that of Griffiths[7] and Ohno.[8] In this step we use the array of values of the z component of the normal to detect and compute silhouette points along the grid lines. If a grid point is different in sign from one of its neighbors, the grid segment joining them contains a silhouette point that is computed by linear interpolation along the grid segment. In grid cells the silhouette is approximated linearly, and two different cases may occur:

1. The cell contour contains two silhouette points. The straight segment joining them approximates the silhouette (Figure 2a).

2. The cell contour contains four silhouette points. The silhouette is approximated by two segments corresponding to one of the cases in Figure 2b—depending on the sign of the z component of the normal in the center of the cell.

The silhouette segments are finally connected to define the silhouettes of the patch. This method is often used in contouring applications.[11]

Let's suppose that silhouettes are connected to the patch boundaries. The patch decomposition is obtained by following its boundary and placing the grid points in a list representing the region's boundary. When a silhouette endpoint is found, the points defining this silhouette are added to the list. The patch boundary is then followed again from the other endpoint of the silhouette until the starting point of the region's boundary is found (see Figure 3).

During generation of the patch's regions, the singular points are computed and placed in lists of singular points belonging to the same scan plane. These lists are sorted in scan direction to quickly update at every scan plane the list of intersected regions. In addition, the set of lists is used to choose exact intersection planes.

Figure 4 shows example results of the different steps in generating the regions.

Scan process

In this section we essentially present the method used by the proposed algorithm to compute the intersection of the patch's regions and scan planes.

Region-exact intersection planes intersection

The intersection of a region and an exact intersection plane consists of two steps:

1. Intersection of the exact intersection plane with the boundary curves of the region.

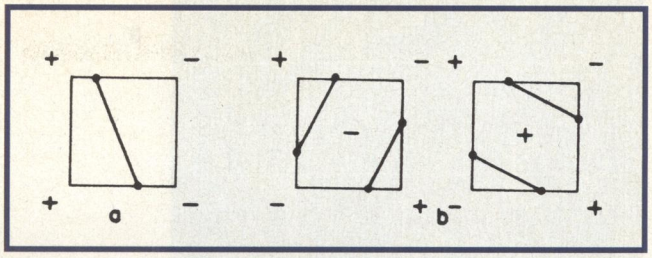

Figure 2. Generation of silhouette segments in a grid cell.

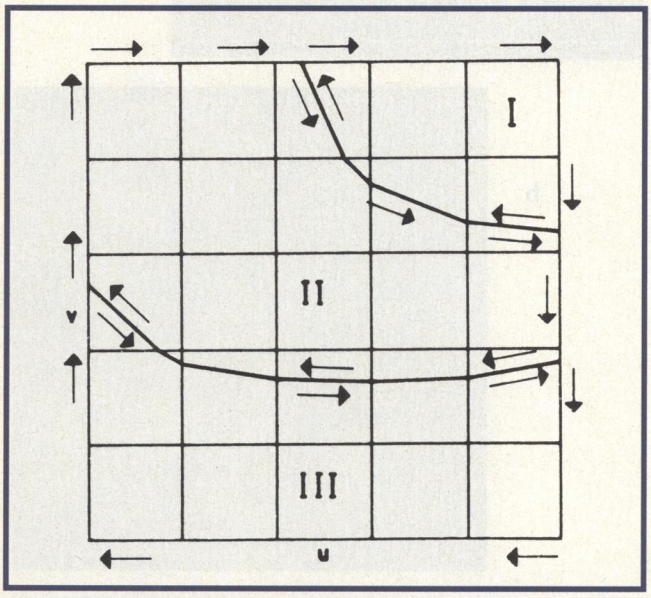

Figure 3. Regions generation process.

2. Segments calculation: connection of the points obtained in step 1 defining the complete intersection.

Boundary intersections

The intersection of a region and an exact intersection plane is used to obtain the intersection of the region and the intermediate planes of the preceding and the following bands. Sometimes these intersections are qualitatively different for these two contiguous bands. In such cases, the exact intersection plane intersection must be different from one band to the other. This intersection is first computed to treat the preceding band, and at this time the exact intersection plane is considered as the current scan plane. When a new exact intersection plane is taken, its predecessor is modified to treat the following band. The way to compute the region's boundary intersections depends on the type of the singular points in the plane.

The presence of a global maximum in the current scan plane is not taken into account, because it has no influence in the preceding band. If the plane contains a global minimum, the intersection is this point.

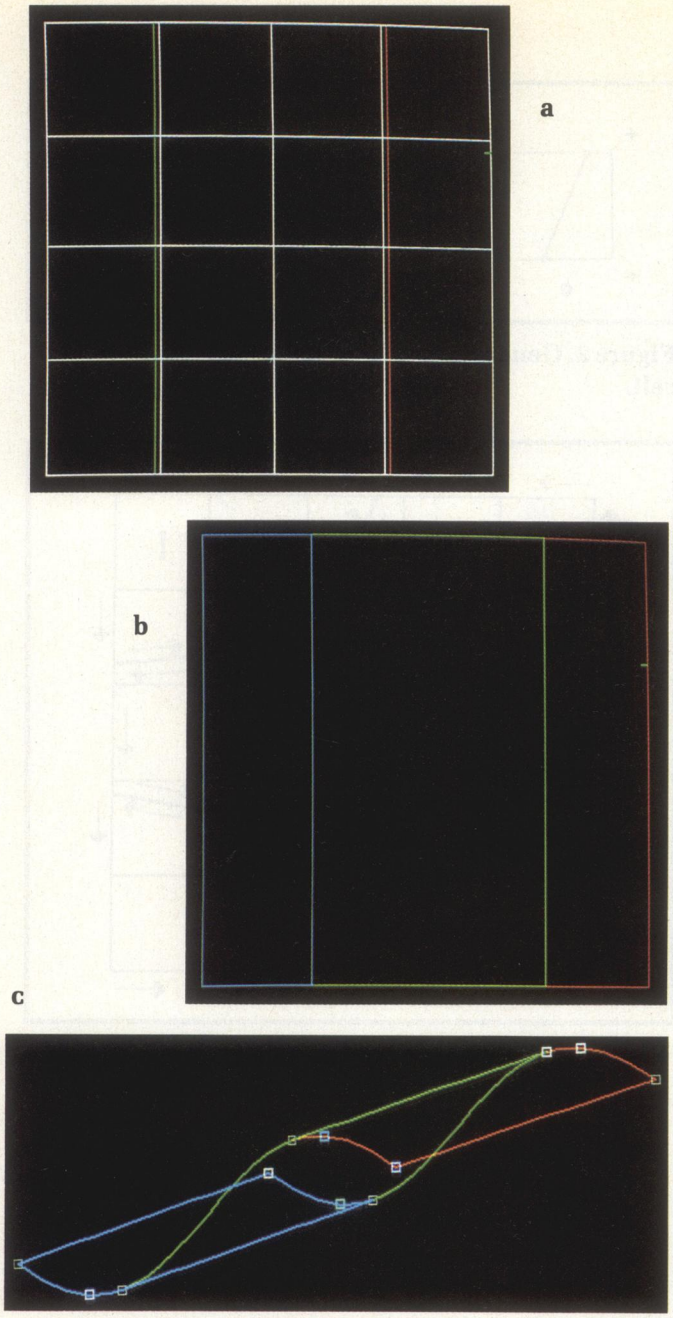

Figure 4. Decomposition process and singular points; (a) silhouettes, (b) regions, (c) singular points.

Next we present the way to compute the current scan-plane intersection in the other cases. The region's boundaries are followed, in the scan direction, starting from the intersection points of the preceding exact intersection plane, until boundary points with the same y coordinate as the current scan plane are found. In fact, this is done in the parametric plane along the list of points defining the region's boundaries. Figure 5 shows the four different cases. When we find a point in this list with a smaller y coordinate than the current plane ordinate, the intersection point is computed by linear interpolation between that point and its precedent. If the intersection is a local minimum or the lowest end of two edges of the region, it is not necessary to follow the boundaries from the preceding exact intersection plane.

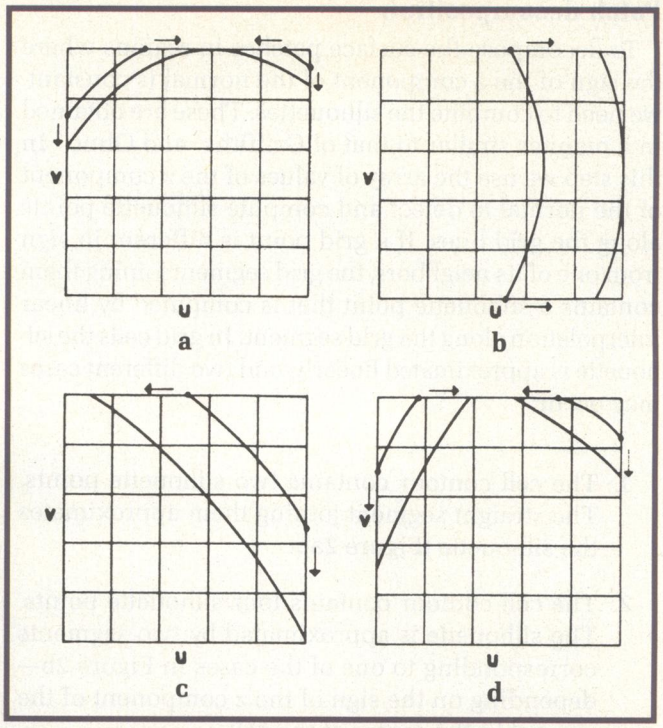

Figure 5. Finding the intersection of an exact intersection plane and the region's boundary.

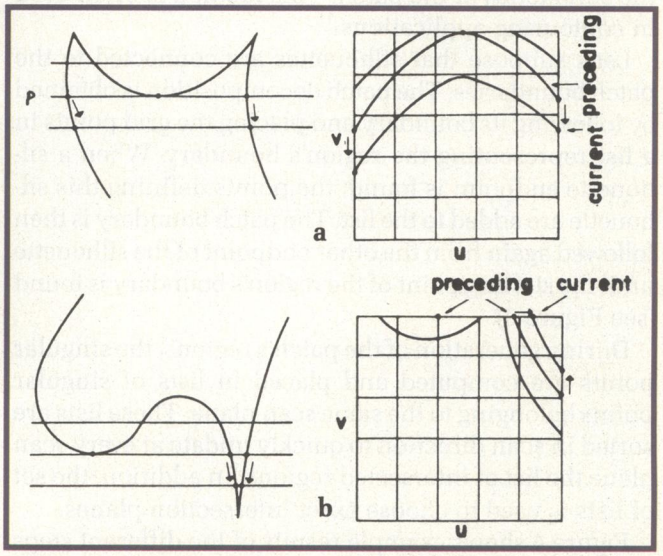

Figure 6. Local minimum cases.

If the preceding exact intersection plane contains a global minimum, the region disappears. If the plane has no singular points, no modification is needed. When the preceding exact intersection plane contains a local minimum, the number of segments of the region decreases. Two different cases might occur, as shown in Figure 6. In Figure 6a the two segments in plane p have a common endpoint. The new intersection will be defined by the two other endpoints. In Figure 6b, we simply remove the zero segment.

The presence of a local maximum in the preceding exact intersection plane has the opposite effect of a local minimum. In Figure 7a, the new intersection will be represented by two segments whose endpoints are the local maximum, and one of the endpoints of the segment defining the intersection before modification. In Figure 7b, a new null-size segment appears. If the preceding exact intersection plane contains a global maximum, the effect is similar to that in the second case of a local maximum.

In most cases, a preceding exact intersection plane with a vertex needs no modification. If the vertex is the lowest endpoint of two edges of the region, it has the same effect as a local minimum. If the vertex is the highest endpoint of two edges, it has the same effect as a local maximum. Table 1 summarizes the effects of different types of singular points in the current and preceding exact intersection planes.

Segments calculation

We have studied[12] two different ways of approximating the segments defined by the intersections of regions' boundaries with an exact intersection plane. One possibility is to approximate, in the parametric plane, a segment by a cubic curve. In this case we compute two points with the same *y* coordinate as the exact intersection plane ordinate, each one belonging to the same grid cell as one of the segment endpoints. These two new points, together with the segment ends, define the cubic approximation. Our study and implementations have shown that this approach is worse than approximating the segment by a polygonal line. Next we look at the second method.

The generation of the polygonal line takes place in the parametric plane by using the mxm grid defined on it. Starting from one of the intersection points of the exact intersection plane and the region's boundaries, we find a new point with the same *y* coordinate along the edges of the grid cell containing the starting point. These two points define the first straight segment of the polygonal line. The new point belongs to two cells: One is the same cell containing the starting point; the other is unexplored. We iterate the previous process in the new cells until we find the cell containing the other endpoint of the segment. By so doing, we generate the polygonal line that approximates the segment (Figure 8). Finally, we compute the (x, z) coordinates and the intensity of the polygonal line's vertices. These values are interpolated linearly in the scan plane.

Region-intermediate plane intersection

The intersection between a region and an intermediate scan plane is obtained by linear interpolation between the intersections of the region and two consecutive exact intersection planes. If the number of vertices in the polygonal lines of these exact intersection planes is the same, we need only interpolate pairs of vertices taken in the *x* direction to obtain the vertices of the polygonal line in the intermediate plane. If the number of ver-

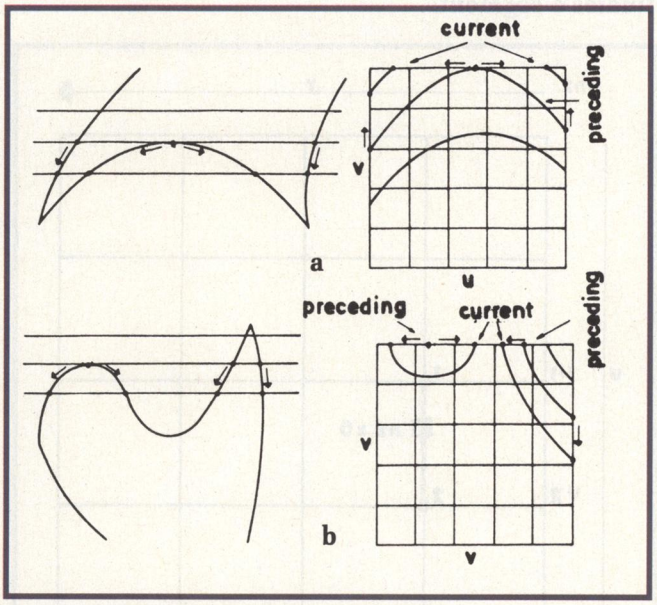

Figure 7. Local maximum cases.

Table 1. Summarizing the effect of different singular points.

SINGULAR POINTS / PLANE	LOCAL MINIMUM	LOCAL MAXIMUM	GLOBAL MINIMUM	GLOBAL MAXIMUM	VERTEX
PRECEDING	Number of segments decrease	Number of segments increase	Disappearance of the region	Appearance of the region	• As with no singular points OR • As a local minimum or maximum.
CURRENT	Simplification of the intersection computation.		Simplification of the intersection computation.		

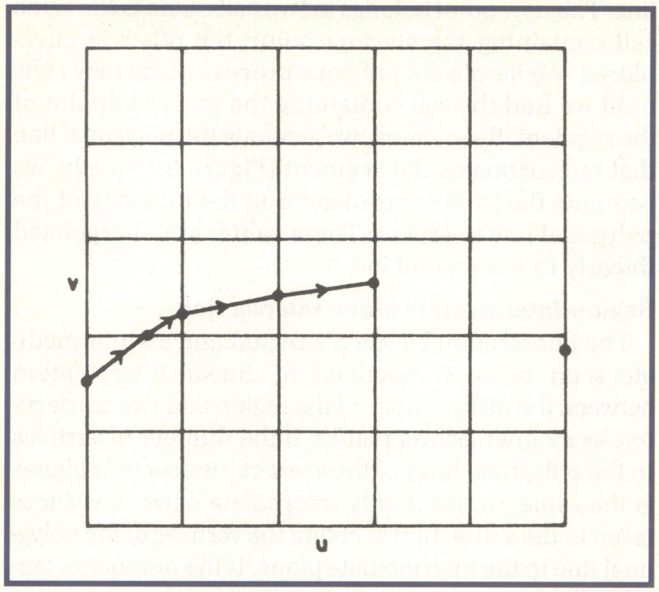

Figure 8. Generation of a polygonal line that approximates a segment.

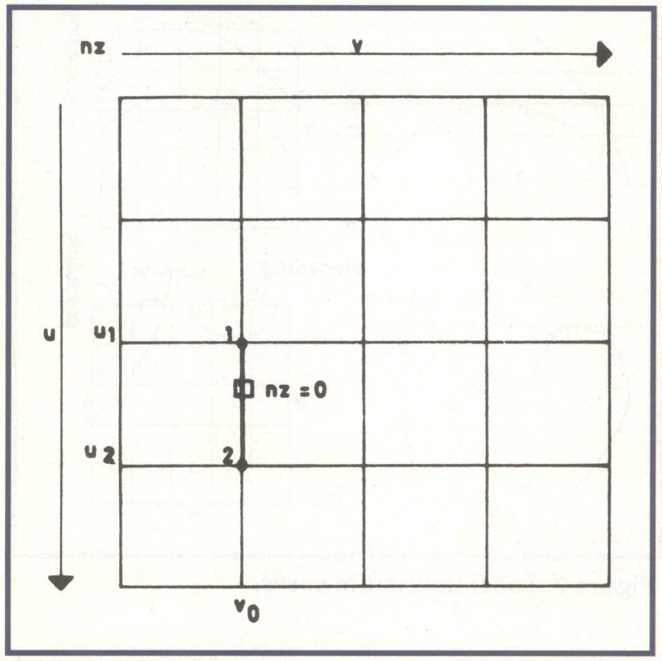

Figure 9. Silhouette points computation by linear interpolation between grid points.

tices in the polygonal lines is different, we add points before the interpolation. This is a typical problem in the reconstruction of objects modeled by a set of parallel sections.[13,14]

Visibility computation

To compute the visibility, we use the scan-line z-buffer technique applied to every straight segment in the polygonal line. Other known methods could be used, taking the polygonal lines as elements of depth comparison. This is possible because there are no silhouette points in the polygonal lines.

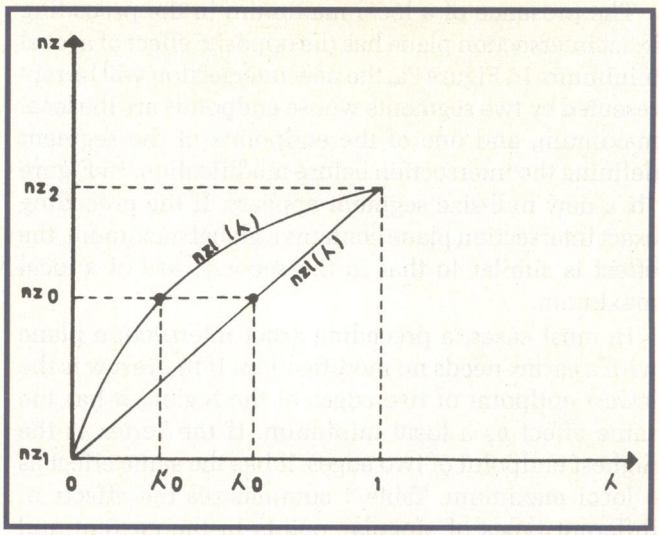

Figure 10. Changing variables and origin in Figure 9.

Error analysis

We now present a study of the errors introduced in the different steps of the proposed algorithm. In the first step, where arrays of the *y* coordinate and the *z* component of the normal are computed, no error is produced.

Silhouette computation error

As we have already seen, we start the silhouette computation by obtaining silhouette points on the grid lines. These points are computed by linear interpolation between two neighbor grid points like 1 and 2 in Figure 9. If the curvature variation of the z component of the normal is not significant in the considered interval, we introduce the change of variable

$$\lambda = (u - u_1) / (u_2 - u_1)$$

to obtain[12] for the isoparametric curve $v = v_0$ the expression

$$nzt (\lambda) = \lambda . f + \lambda . (1 - \lambda) . C \quad (1)$$

where f is the slope of the linear approximation

$$nzl (\lambda) \text{ and } C = \frac{1}{2} (u_2 - u_1)^2 . D^2 nzt (\eta)$$

In this equation (see Figure 10) the difference between the curve (*nzt*) and *nzl* depends on the curve's second derivative, as expected.

Let λ and λ' be the values of the parameter obtained for values of $nz \in [nz_1, nz_2]$ (Figure 10) using the linear approximation and the real curve respectively. We find[12] that

$$\varepsilon = \lambda - \lambda' = \frac{\lambda \cdot (1 - \lambda)}{\omega + 1 - 2 \cdot \lambda} \quad (2)$$

where $\omega = f/C$, and

$$|\varepsilon| < \frac{0.25}{\omega - 1} \quad (3)$$

if $\omega > 1$. From this expression we conclude that ε can be made as small as desired by increasing sufficiently the value of ω. So for a given slope f the error bound is smaller for smaller values of C when $f \neq 0$.

Given that

$$C = \frac{1}{2}(u_2 - u_1)^2 \cdot D^2 nzt(\eta)$$

for any value of $D^2 nzt(\eta)$, we can decrease the value of $u_2 - u_1$ and C by increasing m. So, if $f \neq 0$, it is always possible to find an m value that guarantees an error smaller than any prespecified value. If $f = 0$, we introduce a qualitative error that disappears by changing m.

Equation 3 shows that the error is bound if we can bound $D^2 nzt(u)$. This is possible because we can find[12] an expression of the type

$$D_u^2 nzt(u, v) = [u^5 u^4 u^3 u^2 u\ 1] \cdot B_5 \cdot N_z \cdot B_5^T \cdot [v^5 v^4 v^3 v^2 v\ 1]^T \quad (4)$$

where B_5 is the Bezier array of fifth order. Then, $D_u^2 nzt(u, v)$ will be in the convex hull defined by the N_z components. So the maximum value of N_z will be a bound of $D_u^2 nzt(u, v)$. In the same way we can bound $D_v^2 nzt(u, v)$.

We conclude from this analysis that the error introduced in the silhouette points computation decreases when m rises and that we can easily find a bound for this error.

Region boundary-exact intersection plane intersection error

The error introduced in this step is similar to the one analyzed in the previous section. The only difference is the way to obtain an expression similar to Equation 4 for $D^2 y(u, v)$.

We conclude from our analysis[12] that this is no harder than obtaining the corresponding equation for $D^2 nzt(u, v)$. The error produced in this step decreases[12] when m rises.

Segments computation error

Segments on the exact intersection planes are approximated by a straight line in every crossed grid cell. The vertices of this polygonal line are obtained along the grid lines in the same way as the intersections between exact intersection planes and region boundaries, and the error of this operation is therefore of the same kind. The error produced by the linear interpolation in the grid cells is, again, of the same type as the previously studied errors. Therefore, it decreases when m rises.

Segments on the intermediate planes are computed by linear interpolation between exact intersection planes, and the error in this step is once again similar to the previously studied errors. Therefore, it depends on the surface curvature. Now the error decreases with n, the number of intermediate planes between two consecutive exact intersection planes.

Performance and comparisons

To study the performance of the proposed algorithm implementation, we used the surfaces in Figure 11, where the different types of singular points are present. We have also studied the behavior of the new method when the curvature of the scan-plane elements changes. These changes are obtained by altering the surface orientation.

To quantify the error of the implementation, we compute the exact depth value for 25 points on the surface and their approximate value as obtained by using the proposed algorithm. We define the global error as the average of the square error at each point.

Performance

Tables 2a and 2b show the error values obtained for the surfaces in Figure 11 for different values of the parameters m and n. From these tables we can conclude the following:

1. The influence of parameter n in the global error is small.
2. The influence of parameter m is remarkable. As expected, the error decreases when the value of m grows.

To analyze the influence of the curvature in the error, we rotated the surface in Figure 11a with increments of 0.2 radiants. The following errors were obtained for three consecutive positions: 1.3×10^{-5}, 2.5×10^{-5}, and 3.7×10^{-5}.

Figure 12 shows two images obtained with the proposed algorithm.

Comparisons

Next we compare the proposed method with Whitted's algorithm. This method has been chosen as an element of comparison because it is the most efficient of the "pure" scan-line algorithms,[12] that is, algorithms that compute the intersection between the scene and the scan planes.

The algorithms' complexity

From the cost evaluation of both algorithms, we note the following:

Table 2. Error values of the proposed algorithm for surfaces in Figure 11 and different values of the parameters *m* and *n*.

m \ n	3	5	10	20
4	0.00048	0.00049	0.00052	0.0007
8	0.0002	0.0002	0.0002	0.0004
12	0.000013	0.000015	0.000025	0.000035

a

m \ n	3	5	10	20
4	0.00014	0.00014	0.00014	0.00015
8	0.00003	0.00003	0.00003	0.00006
12	0.000002	0.000002	0.000003	0.000004

b

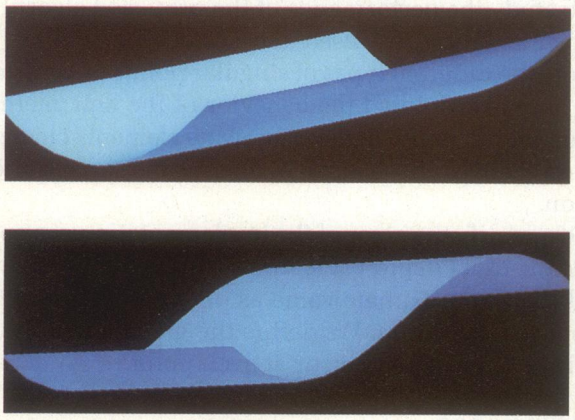

Figure 11. Test surfaces.

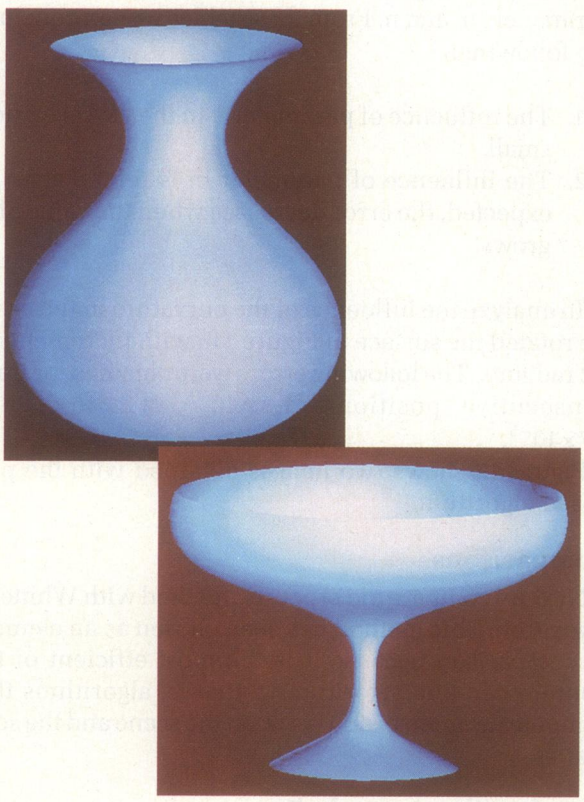

Figure 12. Images obtained with the proposed algorithm.

1. The silhouettes computation is more complex in Whitted's algorithm.
2. The patch subdivision is $0(2^{div})$ in Whitted's algorithm and $0(m^2)$ in the proposed method, where *div* is the subdivision level. So the new algorithm is more expensive in this step.
3. The computation of a scan plane is $0(2^{div} + nbsil)$ in Whitted's algorithm, where *nbsil* is the number of silhouettes in the patch, and $0(m + m.nbsil)$ for an exact intersection plane in the proposed method. The terms 2^{div} and m are the most important in each expression, and they are equivalent. To obtain a similar error, we usually need a bigger value for m than for 2^{div}, and the coefficient of m is greater than the coefficient of 2^{div}. So the computation cost of an exact intersection plane in the proposed algorithm is greater than the computation cost of a scan plane in Whitted's method.
4. The computation cost of an intermediate plane in the proposed algorithm is remarkably lower than the cost of computing a scan plane in Whitted's algorithm.
5. Finally, the visibility computation may be considered of the same order in both algorithms.

The cost of the subdivision is less than that of the scan process. This is proved by our implementations of Whitted's algorithm and the proposed algorithm, where in both cases the scan process is eight times more expensive than subdivision. In the proposed algorithm, most of the scan planes are intermediate planes; therefore, the global cost of the scan process will be less in the proposed method, for most cases. This qualitative remark, plus the fact that the scan process is more important than subdivision, leads us to think that the new algorithm is less expensive than Whitted's. This is verified by the results presented in the next section.

Implementation results

For the surface in Figure 11a and *div* (subdivision level) equals 1, 2, and 3, the global error is 9.6×10^{-4}, 4.3×10^{-4},

Table 3. Performance comparison between the method presented and Whitted's algorithm.

CPU TIME	ERROR: WHITTED	ERROR: NEW ALGORITHM	PARAMETERS: SUB-DIVISION LEVEL	PARAMETERS: m	PARAMETERS: n
11 s	9.6×10^{-4}	3.5×10^{-5}	1	12	20
18 s	4.3×10^{-4}	1.3×10^{-5}	2	12	3

and 1.4×10^{-4}, respectively. Comparing these results with the values in Table 2a, we verify that the error for $div=1$ is greater than every table component. For $div=2$ the error is slightly smaller than the values of the table for $m=4$ and $3\leq n\leq 10$, but is greater than the values corresponding to $m\geq 8$. Finally, for $div=3$ the error is smaller than the values obtained for $m\leq 8$, but is remarkably greater than the errors for $m=12$.

Table 3 presents the error and the parameter values for execution of both algorithms with the same cost. From these results we can conclude that the proposed algorithm is more accurate for the same cost.

The proposed algorithm approximates the scan-plane intersections with more straight segments than Whitted's algorithm, and thus the error is more uniform. This is verified by our implementations. This feature is important because the presence of an unexpected error in one pixel can spoil the image.

Conclusions

The proposed scan-line algorithm for parametrically defined surfaces computes the exact intersection of the surface with only a restricted subset of scan planes. The scan-plane elements are approximated by a polygonal line. We are currently studying the possibility of improving the new algorithm by approximating these elements with curved segments. The accuracy of the proposed algorithm may be improved by obtaining the intermediate planes elements through interpolation in the parametric space, but this modification increases the algorithm's cost. We will also determine the relationship between such an improvement in accuracy and the cost increment.

Essentially, the cost of the proposed algorithm depends on the width of the working bands (parameter n). Its error depends primarily on the accuracy of the parametric grid (parameter m).

For the same cost, the proposed method is more accurate than Whitted's algorithm, and the error is more uniform. ■

Acknowledgments

We would like to thank M. Lucas for useful comments on the algorithm presented and E. Torres for his technical support in its implementation. We are also grateful to the referees for their suggestions.

References

1. J. Lane et al., "Scan Line Methods for Displaying Parametrically Defined Surfaces," *CACM*, Jan. 1980.
2. J. Lane and L. Carpenter, "A Generalized Scan Line Algorithm for the Computer Display of Parametrically Defined Surfaces," *Computer Graphics and Image Processing*, Vol. 11, 1979, pp. 290-297.
3. D. Schweitzer and E.S. Cobb, "Scan Line Rendering of Parametric Surfaces," *Computer Graphics* (Proc. SIGGRAPH 82), July 1982.
4. W. Strasser, *Fast Curve and Surface Display*, doctoral dissertation, Technical University of Berlin, 1974 (in German).
5. A.R. Forrest, "On the Rendering of Surfaces," *Computer Graphics* (Proc. SIGGRAPH 79), Aug. 1979.
6. J.G. Griffiths, "A Data Structure for the Elimination of Hidden Surfaces by Patch Subdivision," *Computer-Aided Design*, July 1975.
7. J.G. Griffiths, "A Surface Display Algorithm," *Computer-Aided Design*, Jan. 1978, pp. 65-73.
8. Y. Ohno, "A Hidden Line Elimination Method for Curved Surfaces," *Computer-Aided Design*, July 1983, pp. 209-216.
9. C. Hornung et al., "An Area-Oriented Analytical Visibility Method for Displaying Parametrically Defined Tensor-Product Surfaces," *Computer Aided Geometric Design*, Sept. 1965.
10. J.G. Griffiths, "A Depth-Coherence Scanline Algorithm for Displaying Curved Surfaces," *Computer-Aided Design*, Mar. 1984, pp. 91-101.
11. P. Brunet, F. Carrasco, and L. Perez, "Automatic Generation of Contour Lines," *Revista Internacional de Métodos Numéricos en Diseño y Cálculo en Ingeniería*, Mar. 1986 (in Spanish).
12. X. Pueyo, *Study of Visualization Algorithms for 3-D Scenes Composed by Curved and Mixed Surfaces Using the Scan Line Principle*, doctoral dissertation, Universitat Politècnica de Catalunya, May 1986 (in Catalan).
13. H. Fuchs, Z.M. Kedem, and S.P. Uselton, "Optimal Surface Reconstruction from Planar Contours," *CACM*, Oct. 1977.
14. R. Thoraval, "Modeling and Visualizing 3-D Objects Described by Sets of Parallel Sections—Commented Bibliography," Tech. Report IMI-info-R18, Université de Nantes, Oct. 1984 (in French).

Xavier Pueyo is a professor with the Computer Science Department at the Polytechnical University of Catalonia, Spain. His research interests include rendering techniques, hidden-surface-removal algorithms, and computer animation. He received an engineering degree from the Polytechnical University of Catalonia in 1980, a PhD in engineering from the same university in 1986, and the degree of Dr. Engineer in computer science from the University of Rennes. He is a member of Eurographics and Associacio Tecnics Informatica, and currently secretary of the Spanish chapter of Eurographics.

Pere Brunet is a professor with the Computer Science Department at the Polytechnical University of Catalonia. He is interested in computer graphics and computer-aided geometric design for both solid models and free-form surfaces. Brunet received an engineering degree from the Polytechnical University of Catalonia in 1970, and a PhD from the same university in 1976. He is a member of Eurographics and ACM, and currently chairman of the Spanish chapter of Eurographics.

The authors can be contacted at Departament de Mètodes Informàtics—ETSEIB, Universitat Politècnica de Catalunya, Av. Diagonal 647, 08028 Barcelona, Spain.