

Decomposing Spline Surfaces into Non-overlapping Regions for Visible Surface Computation*

S. Krishnan

Dept. of Visualization and Display Research
AT&T Shannon Laboratory
Florham Park, NJ 07932, USA
krishnas@research.att.com

D. Manocha

Dept. of Computer Science
Univ. of North Carolina
Chapel Hill, NC 27599, USA
dm@cs.unc.edu

Abstract

Computing the visible portions of curved surfaces from a given viewpoint is of great interest in many applications. It is closely related to the hidden surface removal problem in computer graphics and machining applications in manufacturing. Most of the earlier work has focused on discrete methods based on polygonization or ray-tracing and hidden curve removal. In this paper we present an algorithm for decomposing a given surface into regions such that each region is either completely visible or hidden from a given viewpoint. Initially, it decomposes the domain of each surface based on silhouettes and boundary curves. To compute the exact visibility, we introduce a notion of visibility curves obtained by projection of silhouette and boundary curves and decomposing the surface into non-overlapping regions.

1 Introduction

The problems of visibility and accessibility computations are fundamental for computer graphics, computer-aided design and manufacturing applications. In particular, hidden line and surface removal algorithms in computer graphics are related to visibility computations [FDHF90, Hor84, SSS74, HG77]. Similarly, accessibility computations in manufacturing applications are based on Gauss maps and visibility sets [Woo94, CCW93, GWT94]. These problems have been extensively studied in computer graphics, computer-aided design, computational geometry and manufacturing literature. In this paper, we are dealing with algebraic surfaces and surfaces defined using rational splines [Far93] that are differentiable.

Given a viewpoint, the hidden surface removal problem deals with computation of the surface boundary visible from that viewpoint. Most of the earlier algorithms in the literature are for planar and polygonal primitives and hidden lines removal [FDHF90,

Mul89, SSS74]. In computational geometry literature, many of the hidden surface algorithms simply calculate the entire arrangement of lines (projections of edges and vertices of the objects on the viewing plane). Output-sensitive hidden surface algorithms were developed for special input cases like *c-oriented solids* [GO87], *axis-parallel rectangles* [PVY92] and *polyhedral terrains* [RS88]. Very few algorithms are able to cope with *cycles* (impossible to obtain an ordering among the faces without splitting some of them) efficiently. A randomized algorithm to generate the visibility map was given by Mulmuley [Mul90] for the general case. The algorithm maintains the trapezoidation of the visibility map and updates it by randomly adding one face at a time. The algorithm is (almost) output-sensitive. Extensions of the hidden surface algorithm from planar to curved faces are described in [Mul90]. A survey of most of the recent results in computational geometry regarding object-space hidden surface removal is presented in [Dor94].

When dealing with curved surfaces, most hidden surface removal algorithms must be capable of manipulating semi-algebraic sets [Mul89]. Results from elimination theory and algebraic decision procedures like Gröbner bases are usually used for this purpose [Can88]. Implementations of these algorithms are very non-trivial and applicability of these bounds in practical situations are still not clear.

More recently, a hidden curve removal algorithm has been presented for parametric surfaces by Elber and Cohen [EC90] based on *silhouette curves*. A *silhouette curve* is defined as the loci of points on the surface where the normal vector is orthogonal to the viewing direction. Given a curved surface model and a viewpoint, the silhouettes on the model partition it into front facing and back facing regions (as shown in Figure 1). The surfaces obtained after partitioning based on the silhouette computation need not be completely visible.

We present an algorithm for decomposing a Bézier surface into non-overlapping regions from a given viewpoint. Each Bézier surface is partitioned into non-

*Supported in part by an Alfred P. Sloan Foundation Fellowship, ARO Contract P-34982-MA, NSF Grant CCR-9319957, NSF Grant CCR-9625217, ONR Contract N00014-94-1-0738, ARPA Contract DABT63-93-C-0048 and NSF/ARPA Center for Computer Graphics and Scientific Visualization

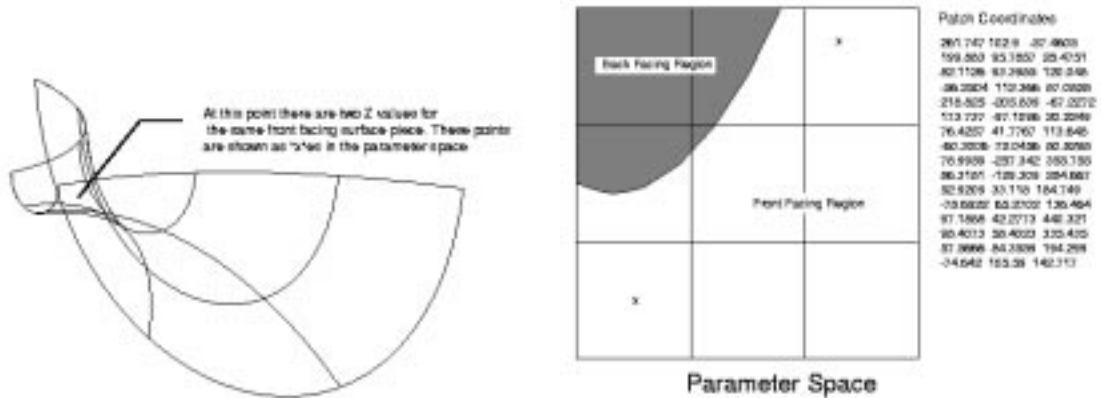


Figure 1: Local Visibility Computations Based on Silhouettes

overlapping regions based on silhouette and *visibility curves* (refer Section 4). Each computed region has the property that it is either entirely visible or invisible in the absence of other surfaces.

In this paper we assume that the input model is composed of non-intersecting surfaces. We have implemented the algorithm for decomposing each surface into non-overlapping regions in finite precision (using 64-bit double precision floating-point arithmetic). The actual performance of the algorithm varies with the viewpoint and surface geometry. On an average it takes about 40-70 milliseconds to decompose one bicubic patch into non-overlapping regions. The main goal of this paper is to present an algorithm for reducing the hidden surface removal problem for spline surfaces to that of polygonal models. Coupled with such an algorithm for polygonal models, we obtain a complete visible surface extraction algorithm for spline surfaces.

The rest of the paper is organized in the following manner. Section 2 briefly describes an overview of our algorithm. We outline an efficient algorithm for computing the silhouettes based on marching methods in Section 3. We introduce the notion of visibility curves in Section 4 and show that silhouettes and visibility curves partition a general surface into non-overlapping regions. We present algorithms and implementations for computation of visibility curves in Section 5. Section 6 talks about how to apply our decomposition algorithm to accomplish hidden surface removal and gives some details about our implementation.

2 Overview of the Algorithm

In this section, we briefly describe the algorithm. The details of individual steps of the algorithm are described in later sections. For the purposes of this paper, we assume that the viewpoint is situated at $z = -\infty$ so that the projection on the viewing plane is orthographic. Given a scene composed of non-intersecting Bézier patches and a viewpoint, we perform the follow-

ing steps for each patch.

- Compute the *silhouette* curves on the patch for the given viewpoint.
- Partition the domain of the Bézier patch as determined by the silhouette curves. The boundary of each partition is made up of the original boundary curves or the computed silhouette curves.
- For each partition so generated,
 - Trace out *visibility* curves (curves in the interior of the partition that have the same projection on the image plane as that of the boundary curves) by following the boundary curves.
 - Partition the domain according to the visibility curves

The set of partitions obtained after executing this algorithm satisfy the property that each partition is non-self-occluding. This fact is proved later in Theorem 2. The main complexity of the algorithm lies in the computation of the silhouette and visibility curves.

3 Silhouettes

Silhouette computation forms an important part of visibility algorithms for curved surfaces. Intuitively speaking, a silhouette curve is the locus of all points on the surface where the normal vector to the surface at that point is perpendicular to the line of sight. We shall restrict our discussion to surfaces whose silhouette (from a given viewpoint) is a curve on the surface. The property of the silhouette curve is that it subdivides the surface into front and back facing regions. However, as shown in Figure 1, silhouettes alone are not sufficient to determine all visible regions. In this section, we describe an algorithm to compute the silhouette curve on a parametric (represented as Bézier [Far93]) patch efficiently.

We assume for the sake of simplicity that the view-point is located at $(0, 0, -\infty)$ and that the view direction is towards positive z -axis. It is easy to see that even if this is not the case, one can always achieve it by applying an appropriate perspective transformation to the control points of the parametric surface $\mathbf{F}(s, t)$. We also require that all the surfaces are differentiable everywhere. We formulate the silhouette curve as an algebraic plane curve in the domain of $\mathbf{F}(s, t)$.

3.1 Formulation of the Silhouette Curve

Let $\mathbf{F}(s, t)$ denote the parametric (differentiable) surface and let $\phi_1(s, t)$, $\phi_2(s, t)$ and $\phi_3(s, t)$ denote the mappings from the parametric space to (x, y, z) space.

$$\mathbf{F}(s, t) = \langle X(s, t), Y(s, t), Z(s, t), W(s, t) \rangle$$

$$\phi_1(s, t) = \frac{X(s, t)}{W(s, t)}, \quad \phi_2(s, t) = \frac{Y(s, t)}{W(s, t)}, \quad \phi_3(s, t) = \frac{Z(s, t)}{W(s, t)}$$

In the rest of the paper, we shall drop the (s, t) suffixes from all the functions for more concise notation. The z -component of the normal at an arbitrary point on the surface is given by the determinant

$$N_z = \begin{vmatrix} \phi_{1_s} & \phi_{1_t} \\ \phi_{2_s} & \phi_{2_t} \end{vmatrix} \quad (1)$$

where ϕ_{i_s} and ϕ_{i_t} denote the partial derivatives of the appropriate function ϕ_i with respect to s and t . For example, $\phi_{1_s} = (WX_s - W_sX)/W^2$ and $\phi_{2_t} = (WY_t - W_tY)/W^2$.

On the silhouette curve, $N_z = 0$. Since $W(s, t) > 0$, we can express the plane curve representing the silhouette as the determinant

$$N_z = \begin{vmatrix} (WX_s - W_sX) & (WX_t - W_tX) \\ (WY_s - W_sY) & (WY_t - W_tY) \end{vmatrix} = 0 \quad (2)$$

Expanding the determinant and rearranging the terms, we can express it as the singular set of the matrix $\mathbf{M}(s, t)$

$$\mathbf{M}(s, t) = \begin{pmatrix} X(s, t) & Y(s, t) & W(s, t) \\ X_s(s, t) & Y_s(s, t) & Z_s(s, t) \\ X_t(s, t) & Y_t(s, t) & Z_t(s, t) \end{pmatrix} = 0 \quad (3)$$

The singular set of $\mathbf{M}(s, t)$ are the values of s and t which make it singular.

3.2 Silhouette Computation

Let us denote the projected silhouette curve corresponding to $\text{Det}(\mathbf{M}(s, t))$ by $D(s, t)$. If the Bézier patch $\mathbf{F}(s, t)$ is of degree m in s and n in t , the curve $D(s, t)$ has degree at most $3(m + n)$. This is a high degree algebraic curve. Our task is to evaluate this curve (*i.e.*, its topological type) completely and efficiently inside our domain of interest.

Our approach is based on marching along the curve using local geometric properties of the curve. All

marching methods require at least one point on every component of the curve inside the domain of interest. We adopt different methods to compute starting points on *open* (intersect the boundary of the domain) and *closed* components (or loops).

To determine starting points on *open* components we substitute one of the variables s or t with the value 0 or 1. This reduces the silhouette equation to a polynomial equation in one variable and this has only a discrete set of solutions. We find all the *boundary silhouette points* by determining the roots of four univariate matrix polynomials, $\mathbf{M}(0, t)$, $\mathbf{M}(1, t)$, $\mathbf{M}(s, 0)$ and $\mathbf{M}(s, 1)$. This problem can be reduced to finding the eigenvalues of an associated companion matrix [MD94]. We retain only the real solutions that lie within the domain.

A much harder problem is to determine if the silhouette curve has loops inside the domain of the surface, and if so to compute at least one point on each of them. We use the fact that the silhouette curve is an algebraic plane curve that is continuous in the complex domain. Since the coefficients of the curve are real, all complex portions of the curve must occur in conjugate pairs. We characterize certain special points on loops (turning points) as places where two complex conjugate paths merge to form a real component. Our idea of loop detection is captured by the following lemma which we state without proof.

Lemma 1 [KM97] *If the curve in the real domain $[0, 1] \times [0, 1]$ consists of a closed component, then two arbitrary complex conjugate paths meet at one of the real points (corresponding to a turning point) on the loop.*

By following all the complex paths inside the domain we can locate at least a single point on each loop. Figure 2 shows the presence of loops in silhouette curves.

Given a point on each component of the silhouette curve, we use marching methods obtain approximations of the next point by taking a small step size in a direction determined by the local geometry of the curve. We have developed an algorithm based on inverse power iterations to *trace* the curve. The details of the complete algorithm are presented in [KM97]. Our algorithm evaluates the silhouette curve at discrete steps to create a piecewise linear approximation. The piecewise linear approximation is computed only to provide an explicit representation of the silhouette curve. However, we do retain the bivariate matrix representation (implicit form) if we need finer approximations of the silhouette curve. The tracing algorithm has been implemented and tested on a variety of examples and has proved to be fairly robust.

3.3 Surface Partitioning based on Silhouettes

Figure 3 shows the silhouette curve on an example patch along with the curve on its domain. It is clear

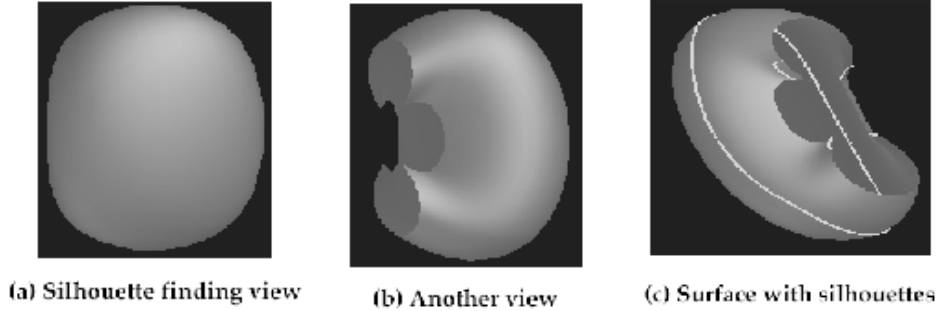


Figure 2: Loops in silhouette curves



Figure 3: Silhouette curve on the patch and the domain

from the figure that the silhouette divides the patch into front and back facing regions.

Lemma 2 *Let $C \subset \mathfrak{R}^3$ be the set of silhouette curves on a given surface $S \subseteq \mathfrak{R}^3$ from a given viewpoint. Then S can be decomposed into C and disjoint regions whose boundaries are surface boundaries and/or parts of C .*

Proof: We omit the proof here because of space restrictions. Please refer to [KM94] for the complete proof.

From the above lemma we can conclude that S can be partitioned into a set of regions R such that the boundaries of each region $\varphi \in R$ consists of original surface boundaries and silhouette curves. The domain of each patch is represented as a simple polygon in counterclockwise order. For example, a complete tensor product Bézier patch has its domain polygon as $\{(0, 0), (1, 0), (1, 1), (0, 1), (0, 0)\}$. In order to subdivide the domain into disjoint regions, we use a stack-based algorithm that traverses along the domain polygon. Details of this algorithm (called *partition_polygon*) can also be found in [KM94].

We now state without proof a fundamental theorem from vector calculus called the *global inverse theorem* which provides the basis for our method [Ful78].

Theorem 1 *Let F be a continuously differentiable mapping defined on an open region $D \in \mathfrak{R}^2$, with range $R \in \mathfrak{R}^2$, and let its Jacobian be never zero in D . Suppose further that C is a simple closed curve that, together with its interior, lies in D , and that F is one-to-one on C . Then the image φ of C is a simple closed*

curve that, together with its interior, lies in R . Furthermore, F is one-to-one on the closed region consisting of C and its interior, so that the inverse transformation can be defined on the closed region consisting of φ and its interior.

The importance of this theorem lies in the fact that properties of the entire region can be argued by looking at the properties of its boundary.

Consider the vector field $M : \mathfrak{R}^2 \rightarrow \mathfrak{R}^2$ such that

$$M(s, t) = (\phi_1(s, t), \phi_2(s, t))$$

where the ϕ_i 's were defined in the previous section. Intuitively, the vector field M projects a point on the surface $F(s, t)$ on to the xy -plane. We now relate the silhouette curve and the Jacobian of the function M .

Lemma 3 *Let M be a mapping from $\mathfrak{R}^2 \rightarrow \mathfrak{R}^2$ such that $M(s, t) = \left(\frac{X(s, t)}{W(s, t)}, \frac{Y(s, t)}{W(s, t)} \right)$. Then the loci of all points on the surface that have vanishing Jacobians corresponds exactly to the silhouette curve.*

Proof: The Jacobian $J(s, t)$ of M is given by

$$J(s, t) = \begin{vmatrix} \frac{(WX_s - W_s X)}{W^2} & \frac{(WX_t - W_t X)}{W^2} \\ \frac{(WY_s - W_s Y)}{W^2} & \frac{(WY_t - W_t Y)}{W^2} \end{vmatrix} \quad (4)$$

However when $J = 0$, we get the same equation as Eqn. (2). That is precisely the equation of the silhouette curve in the domain of the surface. Therefore, the silhouette curves are the only places on the surface where the Jacobian of M vanishes.

We denote the set of regions obtained after partitioning by R . Henceforth, we shall be considering a single element of this set $\varphi \in R$. Because of the previous lemma, we can conclude that the interior of each $\varphi \in R$ have no vanishing Jacobian. Therefore, *global inverse theorem* applies on each of the regions in R .



Figure 4: A helical patch with no silhouettes

4 Visibility Curves

In the previous section, we described a method to compute silhouettes. We shall now introduce the notion of *visibility curves* and elucidate their role in determining visibility.

Consider a region $\varphi \in R$. Let us denote the boundary of φ by $\partial\varphi$ and the interior of φ , the open set $\varphi - \partial\varphi$, by $int(\varphi)$.

Definition 1 Given a region $\varphi \in R$, the visibility curves on φ , $V(\varphi)$, is defined as the locus of points,

$$V(\varphi) = \{p_i | p_i \in int(\varphi), \exists p_b \in \partial\varphi, M(p_i) = M(p_b)\}.$$

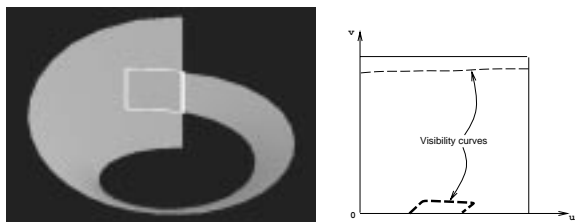


Figure 5: Visibility curves on the helical patch

Intuitively speaking, visibility curves are the loci of all points which lie in the interior of such a partition that have the same projection on the viewing plane as the boundary curves of that partition. It is precisely at these points that the visibility of a patch changes.

Lemma 4 The visibility curves on φ , $V(\varphi)$ as defined above, indeed, form a set of curves.

Proof: Due to space restrictions, we do not include the proof here. See [KM94].

Corollary 1 The set of visibility curves $V(\varphi)$, induces a partition on the region φ .

Proof: Follows from Lemma 2.

Figure 4 shows a helical patch with no silhouettes. Figure 5 shows the visibility curves computed on the same patch. The visibility curves are also shown in the domain of the patch. The patch in Figure 4 is deliberately transformed to provide a better view. Lemma 4 provides a constructive method to find all the visibility

curves. Given the boundary curves of each region, we have to determine all the intersections among the various curves comprising the boundary projections (including self-intersections). We are assuming that the boundary curves are in general position so that their projections intersect only in discrete points. These points form the endpoints for all the visibility curves inside the region φ . Once all the endpoints are found, we use marching methods to trace all the curves. Details of the method are discussed in the next section.

Given $\varphi \in R$, we construct a partition of φ induced by the visibility curves (similar to the one used to obtain partitions induced by silhouette curves). Let us denote the set of regions obtained after partitioning by K_φ . If $k \in K_\varphi$, ∂k denotes the boundary of the region k and $int(k)$, $k - \partial k$, is the interior of k . Figure 6(a) shows the partition of the helical patch based on the visibility curves. In Figure 6(b), we see the image of K_2 under M . It is clear that the boundary of this region is not one-to-one (as seen in the case of points a and a'). However, in the next theorem, we will argue that the interior of each region $k \in K_\varphi$ is one-to-one under the map M .

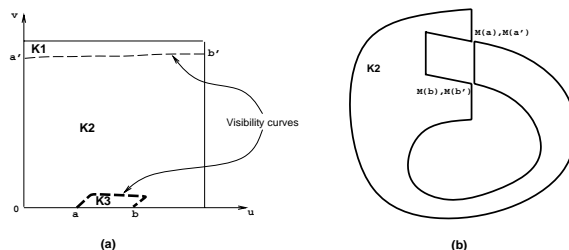


Figure 6: (a) Partitioning based on visibility curves (b) Image of region K_2 under M

Theorem 2 Let $k \in K_\varphi$. Then $int(k)$ is one-to-one under the projection mapping M .

Proof: The main idea of the proof is to show that any simple closed curve C in $int(k)$ is one-to-one under M . Since the Jacobian never vanishes in the interior of each region, we use the *global inverse theorem* to conclude that the entire region on and inside the closed curve C is one-to-one. Since choice of C is arbitrary, we can conclude that $int(k)$ is one-to-one. We would like to re-iterate our earlier assumption that *the boundary curves are in general position so that their projections intersect only in discrete points*.

Consider an arbitrary curve $C \in int(k)$. Assume that C is not one-to-one under the projection mapping M . Therefore, there exists at least two points p_1 and p_2 such that $M(p_1) = M(p_2)$. Consider a simple path (say $path_1$) from p_1 to a boundary point p_{b1} . Since M is a continuous map, there must be a corresponding path ($path_2$) from p_2 . Let us assume that

$path_1$ reaches the boundary point p_{b1} first, and that the point on $path_2$ that has the same projection as p_{b1} is p'_2 . Since $M(p_{b1}) = M(p'_2)$, p'_2 cannot be an interior point (otherwise, it is part of the visibility curve). So p'_2 is also a boundary point. However, since p_{b1} was an arbitrarily chosen point, all its choices must lead to p'_2 s on the boundary. But this contradicts our assumption about boundary curves being in general position. Hence any curve C must be one-to-one under M .

Using C and the *global inverse theorem*, we conclude that the interior of every region $k \in K_\varphi$ is one-to-one under the projection mapping M .

5 Computation of Visibility Curves

In this section, we will describe our method to compute visibility curves. The whole algorithm is split into two parts - (i) finding all the intersections on the projected boundary curves, and (ii) tracing each visibility curve.

5.1 Boundary Intersections

After partitioning based on silhouettes, we obtain regions whose boundaries consist of parts of the original boundary curves of the patch and the silhouette. Let us consider a single region whose boundary is made up of a set of original boundary curves, B , and another set of silhouette curves, L . Each element of B is represented by the corresponding Bézier curve and the interval of parameter values in which it is valid. We also maintain the projection of the boundary curves as a polygonal chain in order to obtain its intersections with silhouette curves. Each element of L and its projection under M is maintained just as a polygonal chain. In order to compute all the intersection points on the projection, we must detect all self-intersections in each element of B and L and intersections between elements. Overall, there are only four basic categories in which all of them fall. We shall discuss each one in detail.

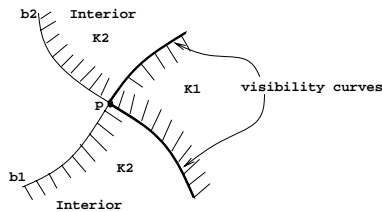


Figure 7: Generation of visibility curves near the boundaries

- 1. Intersection between two boundary curves:** Basically, this case reduces to finding the intersection points between two Bézier plane curves. Let \mathbf{f} and \mathbf{g} be two plane curves parametrized by u and v respectively. The two equations that give rise to

the solution are

$$\frac{X_f(u)}{W_f(u)} = \frac{X_g(v)}{W_g(v)} \text{ and } \frac{Y_f(u)}{W_f(u)} = \frac{Y_g(v)}{W_g(v)}$$

where X_f, Y_f, W_f and X_g, Y_g, W_g are the component-wise functions of \mathbf{f} and \mathbf{g} respectively. By eliminating u from these two equations using Sylvester's resultant [Sed83], we obtain a matrix polynomial in v . We can reduce it to an eigenvalue problem of an associated companion matrix [MD94]. After obtaining all the eigenvalues only the solutions that lie within the intervals are taken. Using this method, all the intersection points are determined accurately and efficiently.

- 2. Self-intersections on boundary curves:** Consider a plane Bézier curve $\mathbf{f} = \langle X(s), Y(s), W(s) \rangle$ of degree n . This curve self-intersects if there exist parameter values u and v , $u \neq v$, such that

$$\frac{X(u)}{W(u)} = \frac{X(v)}{W(v)} \text{ and } \frac{Y(u)}{W(u)} = \frac{Y(v)}{W(v)}$$

Since $u = v$ is a trivial solution to the above pair of equations, we eliminate it by dividing each of the equations by the factor $(u - v)$. Thus the equations become $(X(u)W(v) - X(v)W(u))/(u - v) = 0$ and $(Y(u)W(v) - Y(v)W(u))/(u - v) = 0$. These are two equations each of degree $n - 1$ in u and v . By eliminating u from these equations using Sylvester's resultant, we get a $(2n - 2) \times (2n - 2)$ matrix polynomial of degree $n - 1$. We reduce this problem to one of finding eigenvalues of an associated matrix of size $2(n - 1)^2$. This gives all the self-intersections on the boundary curve.

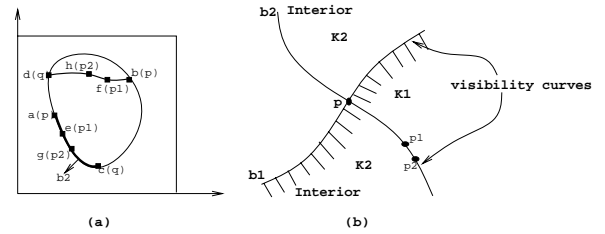


Figure 8: Tracing of visibility curves

- 3. Intersection between two silhouette curves:** Equations governing the solution set are $D(s, t) = D(u, v) = X(s, t)W(u, v) - W(s, t)X(u, v) = Y(s, t)W(u, v) - W(s, t)Y(u, v) = 0$. where $D(., .) = 0$ is the equation of the silhouette curve defined in Section 3.2 and X, Y, Z and W are the component-wise functions of the patch. These four equations in four variables, typically, give rise to a zero dimensional solution set. However, it is

not feasible to solve these four equations directly because of the high degree of the silhouette curve and the algebraic complexity of the resulting system. Therefore, we use piecewise linear approximation of all silhouette curves as the first iteration in our intersection computation. We treat the chain as a set of line segments. Each intersection point obtained is then refined using local minimization methods (we use Powell's method) using the energy equation $E(u, v, s, t)$ given by

$$E(u, v, s, t) = D^2(s, t) + D^2(u, v) + (X(s, t)W(u, v) - X(u, v)W(s, t))^2 + (Y(s, t)W(u, v) - Y(u, v)W(s, t))^2.$$

Self-intersections are also found using the same method. We found that in practice this method of finding intersections works well and gives accurate results.

4. Intersection between boundary curve and silhouette: Consider a boundary curve \mathbf{f} parametrized by u . X_f, Y_f, Z_f and W_f are the scalar functions of \mathbf{F} . X_f , for example, could represent any one of the four functions, $X(s, 0)$, $X(s, 1)$, $X(0, t)$ and $X(1, t)$. The set of intersection points of this curve with the silhouette satisfy $D(s, t) = X(s, t)W_f(u) - W(s, t)X_f(u) = Y(s, t)W_f(u) - W(s, t)Y_f(u) = 0$. In this case, we use the approximated version of the boundary curves (as a piecewise linear chain) and apply a similar procedure as the previous case. The minimization equation in this case is

$$E(u, s, t) = D^2(s, t) + (X(s, t)W_f(u) - X_f(u)W(s, t))^2 + (Y(s, t)W_f(u) - Y_f(u)W(s, t))^2.$$

Once all the intersection points are computed, we are ready to trace all the visibility curves.

5.2 Tracing Visibility Curves

Figure 8 shows the tracing of a visibility curve in the domain of a region and in projection space. Points a and b have the same projection point p . Let the curve $b2$ in Figure 8(b) correspond to the portion of the boundary from a to c (see Figure 8(a)). The boundary of the region on the domain is represented as a polygon (obtained after partitioning based on silhouettes). Let us assume that at an arbitrary step of the tracing method we are at point e on $b2$ and at f on the visibility curve. Both e and f have the same projection point $p1$. Let the domain coordinates of f be (f_u, f_v) . If we move from e to g on $b2$, the point f must move to a neighboring point, h , on the visibility curve. If (g_u, g_v) are the domain coordinates of g , let $x = \frac{X(g_u, g_v)}{W(g_u, g_v)}$ and $y = \frac{Y(g_u, g_v)}{W(g_u, g_v)}$. X, Y and W are the component-wise

functions of the original patch. We would like to find (h_u, h_v) in the local neighborhood of (f_u, f_v) such that

$$\frac{X(h_u, h_v)}{W(h_u, h_v)} - x = \frac{Y(h_u, h_v)}{W(h_u, h_v)} - y = 0$$

Eliminating h_v from these two equations results in a matrix polynomial in h_u . The singular set of this polynomial determines h_u and from the corresponding eigenvector, we can find h_v . However, a lot of unnecessary work can be avoided by observing that (h_u, h_v) is in the neighborhood of (f_u, f_v) . Using f_u as a guess to h_u and building a corresponding eigenvector out of f_v , we perform inverse power iterations [KM97] to obtain (h_u, h_v) .

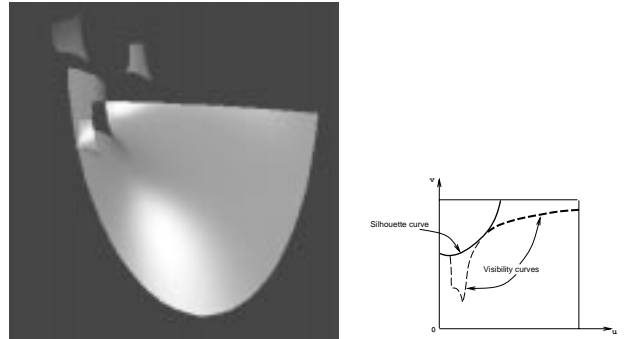


Figure 9: (a) Partitioning of patch in Figure 1 based on visibility curves (b) Visibility curves in the domain

It is possible to use Newton's method to solve the above set of equations. However, as we will see, this method has some problems. Let us assume that point b on the boundary is on a silhouette. We proved that a silhouette point is one where the Jacobian vanishes. Therefore, Newton's method does not perform well close to silhouette points. Inverse power iteration suffers from no such problem.

Tracing terminates when one of the following two cases occur.

- (h_u, h_v) goes out of the region, or
- $(h_u, h_v) = (g_u, g_v)$.

The first case occurs more often. For example, if we trace further from point c , d goes out of the region. Therefore, after each step of the tracing process we have to check for containment of a point inside an arbitrary (simple, but not necessarily convex) polygon. This could be very expensive unless some processing is done on the polygon. We use Seidel's trapezoidation algorithm [Sei91] (created during triangulation) to create trapezoids by horizontal decomposition in $O(N)$ time. Any point can then be determined inside or outside in $O(\log N)$ time.

The second case occurs when the visibility curve hits the boundary curve and then continues along it. This will not be detected in the previous case, and hence, has to be checked explicitly.

After computing all the visibility curves, we calculate all the partitions induced by the visibility curves in each region. Figures 9 and 10 show the partitioning of the patches in Figure 1 and Figure 4 using the visibility curves. Each partition thus obtained is one-to-one under projection. It transforms the original visibility problem into one of n polygon-like surfaces in space. The analysis given so far partitions all parts of a given surface, while only the far regions need to be partitioned. A simple check before tracing visibility can reduce the time and space complexity of the algorithm.

6 Application to Hidden Surface Removal

In the previous sections, we looked at the visibility problem for a single patch. After partitioning each patch based on silhouettes and visibility curves, each region is one-to-one under the projection operation, and can now be treated as a polygon. We shall, therefore, refer to each such region as a *face*. Recently fast randomized algorithms have been developed that can handle this problem for polygonal models [Mul89].

Our final goal is to output trimmed patches of the scene that are visible from the given viewpoint. We shall assume for simplicity that the faces input to this algorithm are non-intersecting. If they are intersecting, we may have to compute all the pairwise surface intersections [KM97] and split them into non-intersecting faces.



Figure 10: Partitioning of patch in Figure 6 based on visibility curves

6.1 Implementation and Performance

The algorithm to compute non-overlapping regions using silhouette and algebraic curves has been implemented. The algorithm uses existing EISPACK [GBDM77] routines for some of the matrix computations. At each stage of the algorithm, we can compute bounds on the accuracy of the results obtained based on the accuracy and convergence of numerical methods adopted like eigenvalue computation, power iteration and Gaussian elimination. Our implementation uses

Model	Degree of patch	Curve computation time (in millisecs.)	Running time (in millisecs.)
Fig. 2	3×3	27.8	53.0
Fig. 3	3×3	23.6	42.4
Fig. 4	1×8	17.3	33.2

Table 1: Performance of our algorithm

EISPACK routines (in Fortran) to compute the eigenvalues of matrices. The algorithm was run on a SGI Onyx workstation with a R4400 CPU with 128 Mbytes of main memory.

We have not implemented the randomized algorithm for performing the general hidden surface removal. Currently, our system takes a set of parametric patches and computes its decomposition into non-overlapping regions. Table 1 shows the performance of our algorithm on certain parametric patches. The time shown for curve computation is the total time for silhouette and visibility curve generation, and the column for running time gives the total time taken by the algorithm for curve generation and producing the non-overlapping partitions.

7 Conclusion

We have presented an algorithm for computing the visible portions of a scene composed of curved (parametric) surfaces from a given viewpoint. We have also given a method to compute the silhouette curve efficiently and correctly. We introduced the notion of visibility curves, which are used to partition each patch into non-overlapping regions. The algorithm has been implemented in floating point arithmetic and performs well in practice.

Acknowledgments

We would like to thank David Banks, David Eberly, Brice Tebbs and Turner Whitted for all the productive discussions and helpful insights. We would also like to thank Brice Tebbs for providing the patch in Figures 1 and 3.

References

- [Can88] J.F. Canny. *The Complexity of Robot Motion Planning*. ACM Doctoral Dissertation Award. MIT Press, 1988.
- [CCW93] L. Chen, S. Chou, and T.C. Woo. Separating and intersecting spherical polygons: computing machinability on three, four and five axis numerically controlled machines. *ACM Transactions on Graphics*, 12(4):305–326, 1993.

- [Dor94] S. E. Dorward. A survey of object-space hidden surface removal. *International journal of Computational Geometry and Applications*, 4:325–362, 1994.
- [EC90] G. Elber and E. Cohen. Hidden curve removal for free form surfaces. *Computer Graphics*, 24(4):95–104, 1990.
- [EC93] G. Elber and E. Cohen. Second order surface analysis using hybrid symbolic and numeric operators. *ACM Transactions on Graphics*, 12(2):160–178, 1993.
- [EC94] G. Elber and E. Cohen. Exact computation of gauss maps and visibility sets for freeform surfaces. Technical report CIS #9414, Computer Science Department, Technion, 1994.
- [Far93] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press Inc., 1993.
- [FDHF90] J. Foley, A. Van Dam, J. Hughes, and S. Feiner. *Computer Graphics: Principles and Practice*. Addison Wesley, Reading, Mass., 1990.
- [Ful78] W. Fulks. *Advanced Calculus: An introduction to analysis*. John Wiley & sons, 1978.
- [GBDM77] B.S. Garbow, J.M. Boyle, J. Dongarra, and C.B. Moler. *Matrix Eigensystem Routines – EISPACK Guide Extension*, volume 51. Springer-Verlag, Berlin, 1977.
- [GO87] R. H. Güting and T. Ottmann. New algorithms for special cases of the hidden line elimination problem. *Comput. Vision Graph. Image Process.*, 40:188–204, 1987.
- [GWT94] J.G. Gan, T.C. Woo, and K. Tang. Spherical maps: Their construction, properties, and approximation. *ASME Trans. J. Mech. Des.*, 1994. To appear.
- [HG77] G. Hamlin and C. W. Gear. Raster-scan hidden surface algorithm techniques. *Computer Graphics*, 11:206–213, 1977.
- [Hor84] C. Hornung. A method for solving the visibility problem. *IEEE Computer Graphics and Applications*, pages 26–33, July 1984.
- [KM94] S. Krishnan and D. Manocha. Global visibility and hidden surface algorithms for free form surfaces. Technical Report TR94-063, Department of Computer Science, University of North Carolina, 1994.
- [KM97] S. Krishnan and D. Manocha. An efficient surface intersection algorithm based on the lower dimensional formulation. *ACM Transactions on Graphics*, 16(1):74–106, 1997.
- [Li81] L. Li. Hidden-line algorithm for curved surfaces. *Computer-Aided Design*, 20(8):466–470, 1981.
- [MD94] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves I: simple intersections. *ACM Transactions on Graphics*, 13(1):73–100, 1994.
- [Mul89] K. Mulmuley. An efficient algorithm for hidden surface removal. *Computer Graphics*, 23(3):379–388, 1989.
- [Mul90] K. Mulmuley. An efficient algorithm for hidden surface removal, ii. Report TR-90-31, Univ. Of Chicago, Chicago, Illinois, 1990.
- [PVY92] F. P. Preparata, J. S. Vitter, and M. Yvinec. Output-sensitive generation of the perspective view of isothetic parallelipeds. *Algorithmica*, 8:257–283, 1992.
- [RS88] J. H. Reif and S. Sen. An efficient output-sensitive hidden-surface removal algorithms and its parallelization. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 193–200, 1988.
- [Sed83] T.W. Sederberg. *Implicit and Parametric Curves and Surfaces*. PhD thesis, Purdue University, 1983.
- [Sei91] R. Seidel. A simple and fast randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry Theory & Applications*, 1(1):51–64, 1991.
- [SSS74] I. Sutherland, R. Sproull, and R. Schumaker. A characterization of ten hidden-surface algorithms. *Computing Surveys*, 6(1):1–55, 1974.
- [TW93] B. Tebbs and T. Whitted. Numerical Design Limited, Personal Communication, 1993.
- [Woo94] T. Woo. Visibility maps and spherical algorithms. *Computer-Aided Design*, 26(1):6–16, 1994.