Geometric Hermite approximation of surface patch intersection curves

Thomas W. Sederberg and Tomoyuki Nishita

Engineering Computer Graphics Lab., Brigham Young University, Provo, UT 84602, USA

Received April 1989
Revised March 1990

Abstract


This paper introduces two new tools for attacking the problem of approximating the intersection curve of two parametric surface patches: the Bézier clipping algorithm for curve/surface intersection, and geometric Hermite approximation of surface/surface intersection curves. The curve/surface intersection algorithm is used to compute a set of endpoint pairs for all components of the intersection curve. Thereafter, a $G^k$ piecewise parametric approximation of the intersection curve is directly computed with no further subdivision or marching. An error bound is determined directly from the approximation. If the error is too large, each unsatisfactory approximating curve is split in half and new approximations are made directly. This procedure is $O(k^{k+2})$ convergent, which means that each time the members of a $G^k$ sequence of approximating curves are split in half, the new error is $2^{-(k+2)}$ of the previous error (in the limit). Thus, doubling the number of approximating curves in a $G^3$ sequence reduces the error typically by two orders of magnitude ($1/256$).

1. Introduction

The problem of computing the intersection curve of two parametric surface patches (hereinafter referred to as an SSI curve for surface–surface intersection) is a fundamental one in computer aided geometric design and graphics. It arises in a broad spectrum of tasks, ranging from constructive solid geometry [Requicha & Voelker '83] to contouring of scattered data [Sabin '85].

It is well known that the algebraic degree of an SSI curve is much larger than the degree of the parametric equations of the intersecting surfaces. For example, two generic tensor product surface patches of parametric degree $m_1 \times m_2$ and $n_1 \times n_2$ respectively, intersect in a curve of degree $4m_1m_2n_1n_2$. Thus, two general bicubic patches intersect in a curve of degree 324. In theory, that curve can be expressed exactly using an implicit equation in terms of the parameters $s$ and $t$ of one of the patches: $f(s, t) = 0$ [Sederberg et al. '84]. In such an implicit equation, the variables $s$ and $t$ each appear to degree 54, so the total degree of the equation is 108.

Far more useful would be a parametric equation $X(u)$ which exactly expresses the SSI curve. Unfortunately, an exact parametric equation for the SSI curve of two bicubic patches in general position is non-existent. This fact is proven by computing the genus of the intersection curve. Only curves of genus zero can be expressed exactly using rational polynomial parametric equations [Salmon '79, p. 30], [Walker '50, p. 67], yet the intersection curve of two general
bicubic patches in general position has genus 433 [Katz & Sederberg '88]. Therefore, SSI curves can, in general, only be approximated using parametric curves.

Two approaches to the problem of approximating SSI curves predominate: subdivision and marching. Subdivision based algorithms (see, for example, [Thomas '84] and [Houghton et al. '85]) characteristically tessellate the surfaces into piecewise linear approximations and intersect the facets. Marching, or curve following, algorithms (see, for example, [Faux & Pratt '83, pp. 257–265], [Geisow '83], [Barnhill et al. '87], and [Bajaj et al. '88]) begin by finding one point on the intersection curve, and proceed to march around the curve, solving three simultaneous trivariate polynomial equations for each successive point.

This paper presents a new solution to the surface–surface intersection problem. Given a beginning and ending point on a branch of the intersection curve, this algorithm directly computes (that is, without any subdivision or marching) two parametric curves (one in the parameter space of each patch) which approximate the intersection curve. An error bound is computed. If the error is unacceptably high, the single approximating curve on each surface patch is replaced with two $G^k$ approximating curves. The error now tends to be $2^{-2k+2}$ times the previous error (in the limit). If the error still isn’t within tolerance, recursive refinement reduces the error by further factors of roughly $2^{-2k+2}$. A pair of approximating curves can be computed in roughly the time needed to perform two steps in a marching algorithm.

This approximation algorithm is based on a technique known as geometric Hermite interpolation—an idea which originates with Sabin [Sabin '68]. Recently, de Boor, Höllig and Sabin described an excellent curve interpolation algorithm based on geometric Hermite interpolation [de Boor et al. '87], and an algorithm for approximate parametrization of planar algebraic curves [Serderberg et al. '89a] has also been developed based on this interpolation strategy. Lee and Fredricks used a similar idea in attacking the plane–surface intersection problem [Lee & Fredricks '84].

This paper also presents a new algorithm for performing curve/surface intersection, an essential component in solving the problem of finding a beginning and ending point on each branch of the intersection curve. Given two rational parametric surface patches, the algorithm begins by subdividing the patches until it is certain that all closed loops have been split. Then, pairs of endpoints for all segments of the intersection curve can be found by intersecting each patch with the boundary curves of the opposing patch.

Section 2 discusses the problem of locating all closed loops in the intersection curve. Section 3 presents a new algorithm for curve/surface intersection. Section 4 describes how to apply geometric Hermite interpolation to the SSI problem and Section 5 discusses error bounds. Section 6 presents an example involving two bicubic patches.

2. Loop detection

Two surface patches may intersect in a complicated curve comprised of several components. In a non-singular intersection (that is, an intersection in which the surface patches are nowhere tangent), there are two types of components, which we will refer to as closed loops and open branches. Fig. 1 shows an intersection curve comprised of one closed loop and one open branch.

The geometric Hermite approximation algorithm requires that we know both endpoints of each segment of the intersection curve that we wish to approximate. It is easy to locate the endpoints on an open branch: simply intersect the boundary curves of each patch with the opposing patch. On the other hand, closed loops have been a major challenge in surface intersection algorithms. Until recently, the detection of closed loops has depended on ad hoc methods (for example, [Timmer '77]) or subdivision/bounding box methods. Ad hoc approaches are not robust, and subdivision methods are expensive.
The problem of loop detection is incidental to this paper. This section surveys the current solutions to the loop detection problem, showing that algorithms are emerging which can robustly locate all closed loops in SSI problems.

Several papers have recently presented new tools for dealing with the loop detection problem. [Sinha et al. '85] presents a criterion for guaranteeing the non-existence of closed loops: If two surfaces intersect in a closed loop, at least one line that is normal (that is, perpendicular) to one surface must be parallel to a line which is normal to the other surface. A stronger theorem is proven in [Sederberg et al. '89b], stating that if two surface patches intersect in a closed loop, if each patch is tangent continuous, and if no normal vector on one patch is perpendicular to any normal vector on either patch, then there exists a line which is simultaneously perpendicular to both surfaces. This is illustrated by the yellow line in Fig. 1. We refer to this as the collinear normal criterion for loop detection, as contrasted to Sinha's parallel normal criterion. We have developed an algorithm, based on the collinear normal theorem and using interval arithmetic, which can locate all collinear normal lines. The algorithm (to be reported in a future paper) converges rapidly and can even find simple point tangencies quickly.

Another interesting new approach to loop detection is based on vector fields [Cheng '89]. This method seeks to locate at least one point on each loop by marching along what are called connecting curves.

[Sederberg & Meyers '88] presents a different loop detection criterion, based on cones which bound constant parameters curves and cones which bound tangent planes: If a cone which bounds all partial derivative vectors on one surface patch lies outside of a cone which bounds all tangent plane directions on the other surface patch, no closed loops exist. This 'cone method' of loop detection can also guarantee that the intersection curve is single valued in one of the patch parameters. Knowing this, if two patches intersect in $n$ open branches, then there is a unique way in which the $2n$ curve endpoints can be connected.

The writeup of the cone method of loop detection in [Sederberg & Meyers '88] warns that the implementation had produced surprisingly slow results in some seemingly well behaved cases. The second author of the current paper found a programming bug in that implementation, fixed it, and demonstrated reasonable response from this algorithm. This loop detection algorithm has been incorporated into the MOVIESTAR.BYU commercial software and performs reliably.

In practice, 'most' pairs of typical surface patches do not intersect in a closed loop, in which case the above surveyed methods can often confirm the non-existence of closed loops with little
effort. These methods also appear to handle correctly patches which intersect in multiple closed loops, even when the radius of the loop is arbitrarily small. Still open for investigation are cases where two patches intersect at a high order point tangency or along a curve of tangency. These cases are particularly vexing in floating point arithmetic.

The geometric Hermite intersection algorithm is independent of the loop splitting method. For the sake of algorithm completeness, we will assume that the cone method is used for loop splitting. A complete algorithm is outlined in [Sederberg & Meyers ’88]. This algorithm returns a set of pairs of subpatches, each pair intersecting in a curve which is at most single valued in one of the patch parameter directions (thus, no closed loops), and the union of the intersection curves of these pairs of subpatches comprises the complete intersection of the original two patches. The remainder of this paper deals with the problem of approximating the curve of intersection of two patches whose interaction has no closed loops.

3. Curve/surface intersection

The geometric Hermite approximation algorithm begins by computing both endpoints of an intersection curve segment. If no closed loops exist, a complete set of endpoint pairs can be obtained by intersecting each patch with the boundary curves of the other patch. This section presents a new, robust algorithm for performing curve/surface intersection for curves and surfaces represented in Bézier form.

This algorithm is based on a concept we will refer to as Bézier clipping. At each step in the iteration, parameter ranges of the curve and/or surface are identified which are guaranteed to not include any intersection points. As the algorithm converges to an intersection point, increasingly large percentages of the remaining curve and surface are trimmed away, so that full double precision accuracy can typically be obtained in four or five iterations, along with a guarantee that all intersection points were located. Bézier clipping has been used previously to compute the intersection of planar algebraic curves [Sederberg ’89], planar parametric curves [Sederberg et al. ’89c], and to perform ray tracing of parametric patches [Nishita et al. ’90].

A rational Bézier surface patch has the equation

$$ P(s, t) = \frac{\sum_{i=0}^{l} \sum_{j=0}^{m} B_i^l(s) B_j^m(t) w_{ij} P_{ij}}{\sum_{i=0}^{l} \sum_{j=0}^{m} B_i^l(s) B_j^m(t) w_{ij}} $$

and a rational Bézier curve is expressed

$$ Q(u) = \frac{\sum_{i=0}^{n} B_i^n(u) v_{ij} Q_{ij}}{\sum_{i=0}^{n} B_i^n(u) v_{ij}} $$

where $B_i^n(t) = (\binom{n}{i}(1-t)^{n-i-1}t^i)$ is the $i$th Bernstein polynomial of degree $n$, $P_{ij} = (x_{ij}, y_{ij}, z_{ij})$ and $Q_{ij} = (x_i, y_i, z_i)$ are three-dimensional control points, and the $w_{ij}$ and $v_{ij}$ are control point weights. We assume all weights to be non-negative so that the convex hull property is valid.

Consider the curve and surface in Fig. 2. The first step is to identify regions of the curve which definitely do not intersect the surface. This is accomplished by finding a parallelepiped (shown in transparency) which bounds the surface reasonably tightly. In our implementation, we choose two of the parallelepiped edges to be parallel to the vectors $\vec{P}_1 = P_{m0} - P_{00}$ and
\( \mathbf{P}_2 = \mathbf{P}_{01} - \mathbf{P}_{00}; \) and the third edge is in the direction \( \mathbf{P}_3 = \mathbf{P}_1 \times \mathbf{P}_2. \) If it happens that \( \mathbf{P}_1 \) is parallel to \( \mathbf{P}_2, \) pick \( \mathbf{P}_3 \) to be any vector perpendicular to \( \mathbf{P}_1. \) The dimensions of the parallelepiped are just large enough to contain all the patch control points.

Denote the equations of the six planes bounding the parallelepiped by \( f_i(x, y, z) = 0, \) \( i = 1, \ldots, 6, \) where the coefficients are signed such that the region bounded by the parallelepiped is \( \{ x, y, z \mid f_i(x, y, z) \leq 0, \ i = 1, \ldots, 6 \}. \) We will refer to \( \{ x, y, z \mid f_i(x, y, z) \leq 0 \} \) as the negative half space of \( f_i, \) and \( \{ x, y, z \mid f_i(x, y, z) > 0 \} \) as its positive half space. Thus, if we can identify a portion of the curve which lies in the positive half space of any plane, that portion of the curve cannot intersect the surface.

The intersection of the curve with plane
\[
ax + by + cz + d = 0
\]
is found by substituting the parametric equation of the curve into the implicit equation of the plane:

\[
\begin{align*}
  a & \left( \sum_{i=0}^{n} v_i x_i B^n_i(u) \right) + b \left( \sum_{i=0}^{n} v_i y_i B^n_i(u) \right) + c \left( \sum_{i=0}^{n} v_i z_i B^n_i(u) \right) + d = 0.
\end{align*}
\]

Clearing the denominator and collecting terms yields
\[
\delta(u) = \sum_{i=0}^{n} \delta_i B^n_i(u) = 0, \quad \delta_i = v_i(ax_i + by_i + cz_i + d).
\]

Note that the distance from the plane to control point \( Q_i = (x_i, y_i, z_i) \) is \( \delta_i / (v_i \sqrt{a^2 + b^2 + c^2}). \)

The function \( \delta(t) \) is a polynomial in Bernstein form, and can be represented as a so-called ‘non-parametric’ Bézier curve [Boehm et al. '84, p. 14] as follows:

\[
D(u) = (u, \delta(u)) = \sum_{i=0}^{n} D_i B^n_i(u).
\]

The Bézier control points \( D_i = (u_i, \delta_i) \) are evenly spaced in \( u \) \( (u_i = i/n). \) Since
\[
\sum_{i=0}^{n} t_i B^n_i(t) = u [(1 - u) + u]^n = u,
\]
the horizontal coordinate of any point \( D(u) \) is in fact equal to the parameter value \( u. \) Fig. 3 shows the curve \( D(t) \) which corresponds to the intersection of the curve with the ‘top’ plane bounding the surface in Fig. 2.
Values of \( u \) for which the curve lies above (that is, in the positive half space of) the top plane in Fig. 2 correspond to values of \( u \) for which \( D(u) \) lies above the horizontal axis in Fig. 3. We can identify parameter ranges of \( u \) for which the curve is guaranteed to lie in the positive half space of the top plane by identifying ranges of \( u \) for which the convex hull of \( D(u) \) lies above the horizontal axis in Fig. 3. In this example, we are assured that \( Q(u) \) lies outside of \( L \) for parameters values \( u < 0.25 \). Thus, the curve is subdivided at parameter value \( u = 0.25 \) and the curve segment over \( 0 \leq u < 0.25 \) is discarded.

This same Bézier clipping procedure is applied to clip the curve against the five other planes. In this example, only the ‘bottom’ plane is able to clip the curve further. It turns out that the bottom plane clips away the curve segment over \( 0.76 < u \leq 1 \).

Once the curve is Bézier clipped against all six planes bounding the patch, the patch is clipped against each of six planes which form a parallelepiped bounding the curve, as shown in Fig. 4. The edge directions of the parallelepiped are taken to be parallel to the vectors \( Q_0 - Q_n \), \( P_{oo} - P_{m0} \) and \( P_{00} - P_{00} \), and the parallelepiped is just large enough to contain all the curve control points. If the angle between any two of those vectors (call them \( A \) and \( B \) and call the third vector \( C \)) is less than, say, 30°, pick the three parallelepiped edge directions to be \( A \), \( C \), and the direction mutually orthogonal to \( A \) and \( C \).

As before, denote the equations of the six planes bounding the parallelepiped by \( f_i(x, y, z) = 0 \), \( i = 1, \ldots, 6 \), where the coefficients are signed such that the region bounded by the parallelepiped is \( \{ x, y, z \mid f_i(x, y, z) \leq 0, \quad i = 1, \ldots, 6 \} \). Now our goal is to identify parameter ranges \( s < s_{\text{min}}, \quad s > s_{\text{max}}, \quad t < t_{\text{min}} \) or \( t > t_{\text{max}} \) for which the surface patch lies in the positive half space of any plane, and to clip away such regions. Obviously, such regions of the surface cannot intersect the curve.

The intersection of the surface with a plane

\[
ax + by + cz + d = 0
\]

is found by substituting the parametric equation of the surface into the implicit equation of the plane. Clearing the denominator and collecting terms yields

\[
\delta(s, t) = \sum_{i=0}^{l} \sum_{j=0}^{m} B_i^l(s) \cdot B_j^m(t) \delta_{ij} = 0,
\]

\[
\delta_{ij} = w_{ij}(ax_{ij} + by_{ij} + cz_{ij} + d).
\]

The distance from the plane to control point \( P_{ij} \) is \( \delta_{ij}/(w_{ij} \sqrt{a^2 + b^2 + c^2}) \).
We now consider the problem of finding ranges of $s$ for which $\delta(s, t) > 0$. The planes most likely to produce the largest clip in $s$ are the two planes defined by the directions $Q_0 - Q_n$ and $P_{00} - P_{m0}$. For this example, we choose the plane nearest our viewpoint in Fig. 4.

The function $\delta(s, t)$ can be represented, in a $(s, t, \delta)$ coordinate system, as a so-called 'non-parametric' Bézier surface (see [Boehm et al. '84]) whose control points $D_j = (s_j, \ t_j, \ \delta_j)$ are evenly spaced in $s$ and $t$: $s_j = i/n, \ t_j = j/m$. A point on such a patch has coordinates

$$D(s, t) = \sum_{i=0}^{l} \sum_{j=0}^{m} B_i(t) \ B_j^m(u) \ P_{ij} = (s, t, \delta(s, t)).$$

The top view of the patch $D(s, t)$ corresponding to the intersection of the patch in Fig. 4 with the front-most plane is shown in Fig. 5.

A side view of the $D(s, t)$ patch, looking down the $t$ axis, is shown in Fig. 6. In this side view, portions of $D(s, t)$ which are completely above the $s$ axis correspond to portions of $P(s, t)$ which lie in the positive half space of the plane. In Fig. 6, the convex hull of the projected control points bounds the projection of the $D(s, t)$ patch. Therefore, we are assured that regions of the $s$ axis which lie completely beneath the convex hull of the projected $D(s, t)$ control points represent portions of $P(s, t)$ which lie in the positive half space of the plane. In this example, that convex hull intersects the $s$ axis at point $s = 0.38$. Thus, we are assured that
Table 1
Parameter ranges for \( P(s, t) \) and \( Q(u) \)

<table>
<thead>
<tr>
<th>Step</th>
<th>([u_{min}, u_{max}])</th>
<th>([s_{min}, s_{max}])</th>
<th>([t_{min}, t_{max}])</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[0.1]</td>
<td>[0.1]</td>
<td>[0.1]</td>
</tr>
<tr>
<td>1</td>
<td>[0.25, 0.76]</td>
<td>[0.38, 0.75]</td>
<td>[0.40, 0.77]</td>
</tr>
<tr>
<td>2</td>
<td>[0.51, 0.59]</td>
<td>[0.56, 0.58]</td>
<td>[0.51, 0.54]</td>
</tr>
<tr>
<td>3</td>
<td>[0.5624, 0.5628]</td>
<td>[0.5667, 0.5668]</td>
<td>[0.5207, 0.5208]</td>
</tr>
</tbody>
</table>

\( Q \) does not intersect \( P \) in the parameter range \( s < 0.38 \). The patch is subdivided to eliminate the region \( s < 0.38 \).

Next, the patch is clipped against the other planes in a similar manner to remove regions \( s > 0.75 \), \( t < 0.40 \), and \( t > 0.77 \). This completes the first iteration in the intersection algorithm. The second iteration proceeds by clipping the remaining curve segment against the planes bounding the remaining surface patch (as shown in Fig. 7) and so forth. The parameter intervals after each step in the algorithm are shown in Table 1. After the fourth iteration, the solution is computed to eight digits of accuracy.

If an iteration fails to reduce the parameter ranges of the surface and of the curve by at least, say, 20\%, there may be more than one intersection. The remedy is to split the curve in half and to compute the intersection of each half with the surface, using a stack data structure to store remaining curve/surface pairs.

4. Geometric Hermite interpolation of SSI curves

The final output of most algorithms for computing SSI curves is a set of piecewise parametric curves which approximate the intersection curve. In the case of marching algorithms, first a number of points are computed on the intersection curve and then the approximating curve is fit through those points. In the case of subdivision algorithms, the intersection curve is approximated by straight line segments, which in turn may be replaced by a fewer number of, say, piecewise cubic curves. By using geometric Hermite interpolation, it is possible to circumvent the computation of points or line segments and directly compute a \( G^k \) piecewise approximation.

4.1. Exact algebraic representation of SSI curves

In theory, it is possible to exactly represent an SSI curve as an implicit equation in the parameter space of either patch. If the two patches are denoted \( P(s, t) \) and \( Q(u, v) \), then by implicitizing one patch and substituting in the parametric equations of the other patch, the intersection curve can be exactly represented \( f(s, t) = 0 \) or \( g(u, v) = 0 \) [Sederberg '83].

Fig. 8 shows two intersecting bicubic patches and Fig. 9 shows their intersection curve in the parameter planes of each patch. For future reference, Fig. 8 also shows the approximating curve created by our algorithm, which can be ignored for our current discussion. The curve is clipped to the \([0, 1]\) parameter limits of each patch. This curve looks innocuous, but it actually is discouragingly complicated. The equation of the intersection curve \( f(s, t) = 0 \) or \( g(u, v) = 0 \) is degree 54 in each variable (total degree 108). Furthermore, the algebraic genus of such a curve is generally 433 [Katz & Sederberg '88], which means there is no hope of an exact parametric equation for the curve.
4.2. Geometric Hermite interpolation

[Sederberg et al. '89a] applies the technique of geometric Hermite interpolation to the problem of parametric approximation of planar algebraic curves. Given two points on an algebraic curve and its implicit equation \( f(x, y) = 0 \), a piecewise \( G^k \) parametric approximation can be found to that curve with relatively little computation. The details of this procedure are documented in [Sederberg et al. '89a], but qualitatively it works like this. Each curve segment in the piecewise sequence is constrained so that at each of its endpoints it intersects the implicit curve with multiplicity \( k + 1 \). Figs. 10(a, b) show a portion of a degree four algebraic curve (thick line width) approximated by one and two \( G^2 \) rational cubic Bézier curves. Each Bézier curve is \( G^2 \) with the implicit curve, and therefore the two Bézier curves are \( G^2 \) with each other (and the error for the two curves is roughly \( 2^{-6} \) times the error for the single curve).

As we contemplate the possibility of applying geometric Hermite interpolation to computing SSI curves, several virtues of the technique assert themselves. For example, the nice convergence rates are a function of \( k \) (the order of geometric continuity) and not of the degree or genus of the algebraic curve being approximated (although the occurrence of inflection points may lower the convergence rate). If the genus of the algebraic curve were zero, then it would be possible to exactly ‘approximate’ it with a single parametric curve, but otherwise the geometric Hermite method works as well on curves of genus 433 as it does on curves of genus 1, and as well on curves of degree 108 as it does on curves of degree 3. Furthermore, and of critical importance, it is not necessary to actually determine the degree 108 equations \( f(s, t) = 0 \) and \( g(u, v) = 0 \) to apply geometric Hermite interpolation. The approximation of the SSI curve can be determined directly from the parametric surface equations \( P(s, t) \) and \( Q(u, v) \) as follows.

Notation. We pause to summarize the notation we have used thus far, and to introduce the notation found in the remainder of the paper:

- \( P(s, t) \) and \( Q(u, v) \): two surface patches whose intersection we wish to approximate.
- \( f(s, t) = 0 \): the exact representation of the SSI curve in the parameter space of \( P(s, t) \).
- \( g(u, v) = 0 \): the exact representation of the SSI curve in the parameter space of \( Q(u, v) \).
- \( S(\alpha) = (s(\alpha), t(\alpha)) \): a parametric approximation to the curve \( f(s, t) = 0 \).
- \( \hat{S}(\alpha) = (\hat{s}(\alpha), \hat{t}(\alpha)) \): a parametrization of the exact curve of intersection \( f(s, t) = 0 \). (Of course, this cannot generally be a rational parameterization.)
- \( U(\beta) = (u(\beta), v(\beta)) \): a parametric approximation to the curve \( g(u, v) = 0 \).
- \( P(\alpha) = P(s(\alpha), t(\alpha)) \): the curve \( S(\alpha) \) mapped to a three-dimensional curve which lies on \( P(s, t) \). Thus, the symbol \( P \) with two parameters indicates the parametric surface, and with one parameter indicates the approximating curve lying on that surface.
- \( \hat{P}(\alpha) \): the curve \( \hat{S}(\alpha) \) mapped to a three-dimensional curve which lies on \( P(s, t) \). Thus, a parametrization of the exact curve of intersection in three space.
- \( Q(\beta) = Q(u(\beta), v(\beta)) \): the image of \( U(\beta) \) on the surface \( Q(u, v) \).
The geometric Hermite approximations to \( f(s, t) = 0 \) and to \( g(u, v) = 0 \) are performed simultaneously, defining two parallel goals:

**Goal 1.** Find an approximating curve \( S(\alpha) \) which is \( G^k \) at both endpoints (\( S(0) \) and \( S(1) \)) with the exact intersection curve \( f(s, t) = 0 \).

**Goal 2.** Find an approximating curve \( U(\beta) \) which is \( G^k \) at both endpoints (\( U(0) \) and \( U(1) \)) with the exact intersection curve \( g(u, v) = 0 \).

One way to accomplish these goals is to require that \( P(\alpha) \) and \( Q(\beta) \) be \( C^k \) at their two endpoints, which is verified through the following two theorems.

**Theorem 1.** If \( S(\alpha) = (s(\alpha), t(\alpha)) \) meets the algebraic curves \( f(s, t) = 0 \) with \( G^k \) continuity in \( (s, t) \) space at \( (s(\lambda), t(\lambda)) \), then \( P(\alpha) \) meets \( \dot{P}(\alpha) \) with \( G^k \) continuity at \( \dot{P}(\lambda) \). Also, if \( P_i(s(\lambda), t(\lambda)) \) and \( P_i(s(\lambda), t(\lambda)) \) are not parallel and if \( P(\alpha) \) meets \( \dot{P}(\alpha) \) with \( G^k \) continuity at \( \dot{P}(\lambda) \), then \( S(\alpha) = (s(\alpha), t(\alpha)) \) meets the algebraic curve \( f(s, t) = 0 \) with \( G^k \) continuity in \( (s, t) \) space at \( (s(\lambda), t(\lambda)) \).

**Proof.** Letting the subscript \( \alpha^k \) indicate the \( k \)th partial derivative with respect to \( \alpha \), we have by the chain rule:

\[
P_{\alpha^k}(\lambda) = P_i(s(\lambda), t(\lambda))s_{\alpha^k}(\lambda) + P_i(s(\lambda), t(\lambda))t_{\alpha^k}(\lambda) + R(P, s(\alpha), t(\alpha)) \tag{3}
\]

where

\[
R(P, s(\alpha), t(\alpha)) = \sum_{i=1}^{k-1} \frac{(k-1)!}{i!} \frac{d^i}{d\lambda^i} \left( P_i s_{(k-i)}(\lambda) \right)
\]

\[
= R(P, s(\alpha), t(\alpha)) \tag{4}
\]

is a collection of terms involving partials \( P_{s^{(i)}(s(\lambda), t(\lambda))} \) with \( i + j > 1 \), \( s_{\alpha^m}(\lambda) \) with \( m < k \), and \( t_{\alpha^m}(\lambda) \) with \( n < k \).

We first show that if \( S(\alpha) = (s(\alpha), t(\alpha)) \) meets the algebraic curve \( f(s, t) = 0 \) with \( G^k \) continuity, then \( P(\alpha) \) meets \( \dot{P}(\alpha) \) with \( G^k \) continuity. Begin by reparametrizing \( S(\alpha) = (s(\alpha), t(\alpha)) \) to be \( C^k \) with \( \dot{S}(\alpha) = (\dot{s}(\alpha), \dot{t}(\alpha)) \) at \( \alpha = \lambda \), which we can do by the definition of geometric continuity. This means that \( S_{\alpha^i}(\lambda) = \dot{S}_{\alpha^i}(\lambda), \) \( i = 0, \ldots, k \). Then, it is clear from the above equation for \( P_{\alpha^i}(\lambda) \) that \( P_{\alpha^i}(\lambda) = \dot{P}_{\alpha^i}(\lambda), \) \( i = 0, \ldots, k \).

It is likewise clear from induction on equations (3) and (4) that if \( P(\alpha) \) meets \( \dot{P}(\alpha) \) with \( C^k \) continuity, then \( S(\alpha) \) meets \( \dot{S}(\alpha) \) with \( C^k \) continuity, given that \( P_i(s(\lambda), t(\lambda)) \) and \( P_i(s(\lambda), t(\lambda)) \) are not parallel. That is, given

\[
P_{\alpha^i}(\lambda) = \dot{P}_{\alpha^i}(\lambda), \quad i = 0, \ldots, k
\]

we must have

\[
s_{\alpha^i}(\lambda) = \dot{s}_{\alpha^i}(\lambda), \quad t_{\alpha^i}(\lambda) = \dot{t}_{\alpha^i}(\lambda), \quad i = 0, \ldots, k.
\]

This is obvious for \( i = 1 \), since \( R(P, s(\lambda), t(\lambda)) = 0 \). It is likewise true for \( i = 2, \ldots, k \), since then

\[
R(P, s(\lambda), t(\lambda)) = R(P, \dot{s}(\lambda), \dot{t}(\lambda)).
\]

**Theorem 2.** If \( P(\alpha) \) and \( Q(\beta) \) meet with \( C^k \) continuity at an endpoint, if \( |P_{\alpha^i}| \neq 0 \) at that point, and if the two patches are not tangent at that point, then \( P(\alpha) \) and \( Q(\beta) \) are at least \( G^k \) with \( \dot{P}(\alpha) \).

**Proof.** If two curves meet with \( C^k \) continuity, they intersect with multiplicity \( k + 1 \). Since curve \( Q(\beta) \) lies on surface \( Q(u, v) \), \( P(\alpha) \) must intersect \( Q(u, v) \) with multiplicity at least \( k + 1 \). Since \( \dot{P}(\alpha) \) lies on both surfaces, it must intersect \( P(\alpha) \) with multiplicity at least \( k + 1 \).
Expressed in equations, sufficient conditions for meeting our two goals are
\[
\frac{\partial \mathbf{P}(0)}{\partial \alpha^i} = \frac{\partial \mathbf{Q}(0)}{\partial \beta^i}; \quad \frac{\partial \mathbf{P}(1)}{\partial \alpha^i} = \frac{\partial \mathbf{Q}(1)}{\partial \beta^i}, \quad i = 0, \ldots, k.
\]

Using subscripts to denote differentiation with respect to the subscript variable, the chain rule produces the following equations for the two endpoints \( \alpha = \beta = 0, 1 \) for up to third order continuity.

\textbf{G}^0 \textit{continuity.} (\gamma = 0, 1)
\[
\mathbf{P}(s(\gamma), t(\gamma)) = \mathbf{Q}(u(\gamma), v(\gamma)). \tag{5}
\]

\textbf{G}^1 \textit{continuity.} (\gamma = 0, 1) The \( G^0 \) condition must be met, along with
\[
\begin{bmatrix}
\mathbf{P}_s(s(\gamma), t(\gamma)) & \mathbf{P}_t(s(\gamma), t(\gamma)) & -\mathbf{Q}_u(u(\gamma), v(\gamma)) & -\mathbf{Q}_v(u(\gamma), v(\gamma))
\end{bmatrix}
\begin{bmatrix}
s_a(\gamma) \\
t_a(\gamma) \\
u_\beta(\gamma) \\
v_\beta(\gamma)
\end{bmatrix} = 0. \tag{6}
\]

\textbf{G}^2 \textit{continuity.} (\gamma = 0, 1) The \( G^1 \) conditions must be met, along with
\[
\begin{bmatrix}
\mathbf{P}_s(s(\gamma), t(\gamma)) & \mathbf{P}_t(s(\gamma), t(\gamma)) & -\mathbf{Q}_u(u(\gamma), v(\gamma)) & -\mathbf{Q}_v(u(\gamma), v(\gamma))
\end{bmatrix}
\begin{bmatrix}
s_{aa}(\gamma) \\
t_{aa}(\gamma) \\
u_{\beta\beta}(\gamma) \\
v_{\beta\beta}(\gamma)
\end{bmatrix}
= -\left(\mathbf{P}_{ss}(s(\gamma), t(\gamma))s_a^2 + 2\mathbf{P}_{st}(s(\gamma), t(\gamma))s_at_a + \mathbf{P}_{tt}(s(\gamma), t(\gamma))t_a^2\right)
+ \mathbf{Q}_{uu}(u(\gamma), v(\gamma))u_\beta^2 + 2\mathbf{Q}_{uv}(u(\gamma), v(\gamma))u_\beta v_\beta + \mathbf{Q}_{vv}(u(\gamma), v(\gamma))v_\beta^2. \tag{7}
\]

\textbf{G}^3 \textit{Continuity.} (\gamma = 0, 1) The \( G^2 \) conditions must be met, along with
\[
\begin{bmatrix}
\mathbf{P}_s(s(\gamma), t(\gamma)) & \mathbf{P}_t(s(\gamma), t(\gamma)) & -\mathbf{Q}_u(u(\gamma), v(\gamma)) & -\mathbf{Q}_v(u(\gamma), v(\gamma))
\end{bmatrix}
\begin{bmatrix}
s_{aaa}(\gamma) \\
t_{aaa}(\gamma) \\
u_{\beta\beta\beta}(\gamma) \\
v_{\beta\beta\beta}(\gamma)
\end{bmatrix}
= -\left(\mathbf{P}_{sss}(s(\gamma), t(\gamma))s_a^3 + 3\mathbf{P}_{ss}(s(\gamma), t(\gamma))s_as_{aa} + 3\mathbf{P}_{st}(s(\gamma), t(\gamma))s_at_a + 3\mathbf{P}_{tt}(s(\gamma), t(\gamma))t_a^3\right)
+ 3\mathbf{P}_{ss}(s(\gamma), t(\gamma))s_{aa}t_a + 3\mathbf{P}_{st}(s(\gamma), t(\gamma))s_a^2t_a + \mathbf{P}_{tt}(s(\gamma), t(\gamma))t_a^2
+ 3\mathbf{P}_{st}(s(\gamma), t(\gamma))s_{aa}t_a + 3\mathbf{P}_{tt}(s(\gamma), t(\gamma))t_a^3
+ \mathbf{Q}_{uu}(u(\gamma), v(\gamma))u_\beta^3 + 3\mathbf{Q}_{uv}(u(\gamma), v(\gamma))u_\beta u_\beta v_\beta
+ 3\mathbf{Q}_{uv}(u(\gamma), v(\gamma))u_\beta^2 v_\beta + 3\mathbf{Q}_{vv}(u(\gamma), v(\gamma))v_\beta^3
+ 3\mathbf{Q}_{uv}(u(\gamma), v(\gamma))u_\beta v_\beta^2 + 3\mathbf{Q}_{vv}(u(\gamma), v(\gamma))v_\beta v_\beta. \tag{8}
\]
The solution to equation (5), \( s(0), t(0), u(0), v(0) \) and \( s(1), u(1), t(1), v(1) \), is arrived at initially by solving the curve–surface intersection problem using the algorithm discussed in Section 2. The curves involved are typically patch boundary curves.

If, after we have found our initial pair of approximating curves, they need to be split in order to attain better precision, then equation (5) is solved by performing Newton iteration directly on the two surfaces—a standard procedure in any marching algorithm [Bajaj et al. '88]. The starting guess is the parametric midpoints, \( P(s(0.5), t(0.5)) \) and \( Q(u(0.5), v(0.5)) \). This has not failed in our experience. However, it is possible that a convergence problem could arise if the error between the two approximating curves is excessive. In such a case, one could resort to the curve/surface intersection algorithm to solve equation (5) as follows. Suppose the cone test for loop detection had revealed that the intersection curve is single valued in \( s \). Solve equation (5) for the two new approximating curves by intersecting surface \( Q \) with the isoparameter curve \( P(\lambda, \tau) \) where \( \lambda = (s(0) + s(1))/2 \) on the initial approximating curve \( s(\alpha) \).

The continuity equations (6)–(8) are linear in the unknowns. Thus, given \( S(\gamma) = (s(\gamma), t(\gamma)) \) and \( U(\gamma) = (u(\gamma), v(\gamma)) \) (endpoints of the curve segments), one can compute \( S_a(\gamma), S_u(\gamma), S_t(\gamma), S_{uu}(\gamma), S_{uu}, S_{vu} \) etc. by simply solving some sets of linear equations. Note that each of these equations has four unknowns but only three scalar equations (for the \( x, y \) and \( z \) components respectively). A suggestion on what to do with the extra degree of freedom is made in subsection 4.3.

It is convenient for computing error bounds to express the curves \( S(\alpha) \) and \( U(\beta) \) as Bézier curves, which is easily done once equations (5) and (6) have been solved (along with equations (7) and/or (8), if desired). For \( G^k \) continuity, it is simplest to use polynomial curves of degree \( 2k + 1 \); there is always a solution and each control point can be solved from a single linear equation. The control point \( S_0, S_1, \ldots, S_{2k+1} \) for the Bézier approximating curve \( S(\alpha) \) can be solved from the derivative formulae for Bézier curves as follows:

\[
S_0 = S(0); \quad S_{2k+1} = S(1).
\]

For \( k \geq 1 \):

\[
S_1 = S(0) + S_u(0)/(2k + 1); \quad S_{2k} = S(1) - S_u(1)/(2k + 1).
\]

For \( k \geq 2 \):

\[
S_2 = 2S_1 - S_0 + S_{uu}(0)/(2k(2k + 1));
\]

\[
S_{2k-1} = 2S_{2k} - S_{2k+1} + S_{uu}(1)/(2k(2k + 1)).
\]

For \( k \geq 3 \):

\[
S_3 = 3S_2 - 3S_1 + S_0 + S_{uu}(0)/((2k - 1)2k(2k + 1));
\]

\[
S_{2k-2} = 3S_{2k-1} - 3S_{2k} + S_{2k+1} - S_{uu}(1)/((2k - 1)2k(2k + 1)).
\]

We assume that neither surface patch is singular (no cusps or self-intersections). This is reasonably for patches in computer aided geometric design. Equations (5)–(8) also assume that the intersections are transversal (no points of tangency). See [Sederberg et al. '89a] for details on how to extend the geometric Hermite method to handle simple tangencies. The problem of higher order tangencies is an open question, especially in the context of floating point arithmetic.

There may be some confusion about our use of the terms geometric continuity and parametric continuity. In equations (6)–(8), we actually solve for the parametric continuity between \( P(\alpha) \) and \( Q(\beta) \). However, since the actual curve of intersection is not parametric, it is not proper to say that the approximating curves are parametrically continuous. Furthermore, neighboring curve segments \( P(\alpha) \) will in general have geometric continuity, but not parametric continuity.
4.3. Tutorial example

We now step through a simple example, computing a $G^1$ approximation to the intersection of the two bilinear surfaces shown in Fig. 11. The two surfaces are

$$P(s, t) = P_{00}(1 - s)(1 - t) + P_{10}s(1 - t) + P_{01}(1 - s)t + P_{11}st$$

where $P_{00} = (0, 0, 0)$, $P_{10} = (0, 1, 4)$, $P_{01} = (3, 3, 0)$, and $P_{11} = (4, 0, 4)$ and

$$Q(u, v) = Q_{00}(1 - u)(1 - v) + Q_{10}u(1 - v) + Q_{01}(1 - u)v + Q_{11}uv$$

where $Q_{00} = (0, 0, 0)$, $Q_{10} = (0, 4, 4)$, $Q_{01} = (4, 2, 0)$, and $Q_{11} = (4, 0, 4)$. By inspection, equation (5) (for $\gamma = 0, 1$) is satisfied by setting

$$S(0) = (s(0), t(0)) = (0, 0); \quad S(1) = (1, 1);$$

$$U(0) = (u(0), v(0)) = (0, 0); \quad U(1) = (1, 1).$$

Equation (6) (with $\gamma = 0$) evaluates to:

$$\begin{bmatrix}
0 & 3 & 0 & -4 \\
1 & 3 & -4 & -2 \\
4 & 0 & -4 & 0
\end{bmatrix}
\begin{bmatrix}
s_{\alpha}(0) \\
t_{\alpha}(0) \\
u_{\beta}(0) \\
v_{\beta}(0)
\end{bmatrix} = 0. \quad (9)$$

The solution to this set of homogeneous linear equations is only unique to a scale factor, so we must place one more constraint on the equations. If $f(s, t) = 0$ is single valued with respect to the direction perpendicular to $S(0) - S(1)$ in the $(s, t)$ parameter plane, then a reasonable constraint is

$$(S(1) - S(0)) \cdot S_{\alpha}(\gamma) = (S(1) - S(0)) \cdot (S(1) - S(0)), \quad \gamma = 0, 1.$$

This makes the curve a single valued function with respect to the direction perpendicular to $S(0) - S(1)$ in the $(s, t)$ plane. For solving second (and higher) derivatives, the constraint is

$$(S(1) - S(0)) \cdot S_{\alpha\alpha}(\gamma) = 0.$$

Getting back to our example, we impose the constraint

$$(1, 1) \cdot (s_{\alpha}, t_{\alpha}) = (1, 1) \cdot (1, 1)$$

Fig. 11. Two bilinear surfaces.
or \( s_\alpha(0) + t_\alpha(0) = 2 \) and solve

\[
s_\alpha(0) = 2/3; \quad t_\alpha(0) = 4/3; \quad u_\beta(0) = 2/3; \quad v_\beta(0) = 1.
\]

Equation (6) for \( \gamma = 1 \) is

\[
\begin{bmatrix}
1 & 4 & 0 & -4 \\
-3 & -1 & 2 & 4 \\
4 & 0 & -4 & 0
\end{bmatrix}
\begin{bmatrix}
s_\alpha(1) \\
t_\alpha(1) \\
u_\beta(1) \\
v_\beta(1)
\end{bmatrix} = 0. \quad (10)
\]

Again imposing the constraint \( s_\alpha(1) + t_\alpha(1) = 2 \), we solve

\[
s_\alpha(1) = 2; \quad t_\alpha(1) = 0; \quad u_\beta(1) = 2; \quad v_\beta(1) = 1/2.
\]

These endpoint derivatives define Bézier curves \( S(\alpha) \) with control points \( S_0 = (0, 0), S_1 = (2/9, 4/9), S_2 = (1/3, 1), S_3 = (1, 1) \) and \( U(\beta) \) with control points \( U_0 = (0, 0), U_1 = (2/9, 1/3), U_2 = (1/3, 5/6), U_3 = (1, 1) \). These two curves, are shown in Fig. 12 with thick line width, and the exact intersection is shown in thin line width.

5. Error estimation

We measure the error as the maximum distance from \( P(\alpha) \) to \( Q(\beta) \). An estimate of this error can be obtained by computing the distance \( \| P(\alpha) - Q(\beta) \| \) for several values of \( \alpha = \beta \).

Happily, an error bound can also be obtained if the two surfaces are polynomial parametric. Note that \( P(\alpha) \) can be expressed as a three-dimensional Bézier curve (with control points \( P_i \)) by composing the curve \( S(\alpha) \) with the surface \( P(s, t) \) (likewise for the curve \( Q(\beta) \)). If the degree of \( S(\alpha) \) is \( d \) and if \( P(s, t) \) is tensor product degree \( m \times n \), then the degree of \( P(\alpha) \) is \( d(m + n) \). If \( P(s, t) \) is triangular and degree \( n \), the degree of \( P(\alpha) \) is \( dn \). For a discussion of the composition of Bézier curves and surfaces, we refer the reader to [DeRose '88]. A bound on
the error is then simply the largest distance $P_i - Q_i$, which can be seen as follows:

$$\max_{0 \leq \alpha \leq 1} \left[ \min_{0 \leq \beta \leq 1} \| P(\alpha) - Q(\beta) \| \right] \leq \max_{0 \leq \alpha \leq 1} \| P(\alpha) - Q(\alpha) \|$$

$$= \max_{0 \leq \alpha \leq 1} \left\| \sum_{i=0}^{k} (P_i - Q_i) B_i^k(\alpha) \right\|$$

$$\leq \max_{i=0, \ldots, k} \| P_i - Q_i \|$$

(11)

where $k$ is the degree of $P(\alpha)$ and $Q(\beta)$. If they are initially of different degree, degree elevate to make them the same degree.

In our example, the curves $P_i$ and $Q_i$ are degree six. In Fig. 13, $P_i$ is shown in yellow (with yellow control points) and $Q_i$ is shown in red (with red control points). Note that since these curves have $C^1$ continuity, $P_0 = Q_0$, $P_1 = Q_1$, $P_5 = Q_5$, and $P_6 = Q_6$. A bound on the error is then $\max(\| P_2 - Q_2 \|, \| P_3 - Q_3 \|, \| P_4 - Q_4 \|)$.

While this is a guaranteed bound of the max-min distance between $P(\alpha)$ and $Q(\beta)$, it does not necessarily bound the distance from those curves to the actual intersection curve. If the surfaces meet at a shallow angle, $P(\alpha)$ and $Q(\beta)$ may be much closer to each other than to the actual intersection curve.

The error bound calculation favors low degree approximating curves, because the degree of $P(\alpha)$ can get large. Bicubic patches with cubic approximating curves produce $P(\alpha)$ of degree 18. An example of this is shown in Fig. 8. The error bound also works for rational patches, although in that case it does not suffice to compare distances between control points. Instead, rational functions must be differenced ($P(\alpha) - Q(\alpha)$ in equation (11)), for which it is expensive to obtain a tight error bound.

The maximum error bound often occurs near the parametric midpoints of the curves. Therefore, a tighter error bound can be obtained by subdividing $P(\alpha)$ and $Q(\beta)$ and comparing distances between the control points of corresponding curve halves.

6. Bicubic example

We now illustrate the algorithm with an intersection example involving two patches from Newell’s teapot [Crow '87]: a patch from the spout and a patch from the body (see Fig. 13). Figs. 14–17 show the error involved in approximating the curve using different degrees of continuity ($k$) and different number of piecewise curve segments. The error is color coded, with red indicating two digits of accuracy and blue indicating ten digits of accuracy, relative to the width of the spout patch.

In Fig. 14, the intersection is approximated by $G^0$ curves, which happen to be merely straight lines in the parameter spaces of the respective patches. In all cases, the endpoints of the curve segments are accurate to at least 10 digits (hence the blue color). Even after splitting the approximation into 16 segments, the $G^0$ piecewise approximation attains only two digit accuracy. This is similar to the error attained from the subdivision method of surface intersection, though far fewer subdivisions are required.

Fig. 15 shows a $G^1$ approximation, for which we used cubic curves in parameter space. As can be seen, two $G^1$ segments provide three digit accuracy in this example, and eight segments provide six digit accuracy. Fig. 16 shows that $G^2$ approximation in this example produces three, five, six and eight digit accuracy respectively for one, two, four and eight segments. Finally, Fig. 17 shows that merely two $G^3$ approximating segments provide six digits accuracy and eight segments provide ten digits accuracy in this example.
The error convergence in this example is typical of all non-singular intersection curves. In the limit, a $G^k$ piecewise approximation yields $O(h^{2k+2})$ error convergence (see, for example, [de Boor et al. '87] or [Davis '75]). This means that, in the limit, each time the curve segments in a $G^k$ approximation are split in half, the error reduces by a factor of $2^{-2k+2}$ in the limit. This convergence rate is nicely illustrated in Figs. 14–17.

The patches in Fig. 1 were preprocessed using the cone test for detecting closed loops. This resulted in an initial piecewise approximation to the intersection curve involving 12 curve segments as shown. The maximum error is three digits of accuracy for these 12 curves. The approximating curves are $G^1$ cubic Bézier curves. As suggested in Figs. 14–17, the error can be quickly reduced to any tolerance by splitting the approximating curves.
6.1. Higher orders of continuity

One may postulate from Figs. 14–17 that as $k$ gets large, a single curve segment might approximate the intersection curve to any desired accuracy. This holds true in some cases, but it depends on the radius of convergence of the power series at each endpoint, something which is difficult to compute analytically. For small radii of convergence, approximating curves with larger $k$ may actually diverge from the intersection curve, producing larger errors than curves with lower $k$.

7. Conclusions

We have presented algorithms on which to base a surface/surface intersection package. Most of these algorithms have been implemented in the MOVIESTAR.BYU commercial software package, and perform well. We are confident that enough detail is provided in this paper to allow basic implementation. Trial and error will dictate some fine tuning considerations.

The geometric Hermite intersection algorithm can be applied to any pair of parametric surfaces for which derivative values can be computed. The loop detection algorithm can be applied to any pair of surfaces for which bounds of first derivatives can be determined. The error bound is primarily useful for polynomial surfaces.

The geometric Hermite approximation algorithm is impressive in its precision, and appears to perform quickly. We are planning to perform extensive tests to compare its speed and precision with those of alternative algorithms.

Open questions include how to deal with tangent intersections of arbitrary complexity, and floating point error analysis.

Acknowledgement

This work was supported in part by the National Science Foundation under grant number DMC-8657057. The authors acknowledge valuable discussions with Geng-Zhe Chang, Alyn Rockwood, Wayne Tiller, K.-P. Cheng, Ron Goldman, and some useful comments from the referees. Alan Zundel helped create some of the figures.

References

Davis, P.J. (1975), Interpolation and Approximation, Dover, New York.
Thomas, S.W. (1984), Modeling volumes bounded by B-spline surfaces, Ph.D. Dissertation, Department of Computer Science, University of Utah.