1995-07-22

Surface Intersection Loop Destruction

Thomas W. Sederberg
tom@cs.byu.edu

Alan K. Zundel

Original Publication Citation

A. K. Zundel and T. W. Sederberg, (aka surface intersection loop destruction)"Loop destruction using Bézier clipping," The Mathematics of Surfaces VII, R. R. Martin, ed., Oxford University Press, 1996, pp. 463-478.

Surface Intersection Loop Destruction

Alan K. Zundel and Thomas W. Sederberg
Brigham Young University

22 July 1996

1 Introduction

The intersection curve between two surface patches consists of one or more connected components or *branches*. Each component can be classified as either an open branch, with endpoints on at least one patch boundary, or as a closed *loop* (see Figure 1).

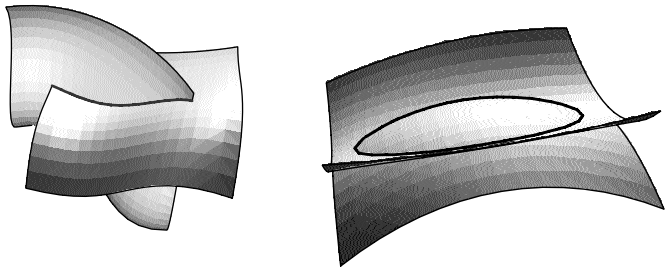

Figure 1: Open Branch and Closed Loop

It is known that the intersection curve formed by two rational surface patches is generally of very high degree (324 in the case of two bicubic patches) and genus (433 in the case of two bicubic patches) [KS88]. The large genus means it is impossible to exactly express the intersection curve in the form of a parametric equation. Hence, algorithms for "computing" an intersection curve usually are designed to compute a piecewise approximation of the curve to within a tolerance.

Most such algorithms require that an initial point be identified on each branch of the

intersection, and then a curve-following scheme is invoked to compute the approximation [Pra86, BHHL88, CO88, BK90, SN91]. Whereas starting points on open branches can be found by simply intersecting each surface with the boundary curves of the opposing surface, starting points on loops can be more elusive.

This paper introduces an algorithm for "loop destruction" — intelligently subdividing a pair of intersecting Bézier surface patches so as to slice through the closed loops, thereby converting them into open branches. The algorithm is based on a technique called *Bézier clipping* — a variation of interval arithmetic that takes advantage of the convex-hull property of Bernstein polynomials. Loops are typically split in one or two tries if their diameters are at least, say, one-eighth the width of the patch. For infinitesimally small loops, the method converges quadratically.

The algorithms presented here can be applied to any surface type for which the tangent directions can be computed. Our development focuses on Bézier tensor product surface patches.

Various methods have been proposed for detecting closed loops. A simple but fallible method is to extract a set of isoparametric curves from one surface and intersect them with the opposing surface [Tim77]. This algorithm detects big loops multiple times, but may miss small loops. More robust loop detection techniques have been developed based on various strategies such as tesselation and subdivision techniques (see, for example [Arn87, LR80, Dok85, HAG83, HEFS85]); surface, tangent and normal cones [SM88, Hoh91, PP90]; parallel and collinear normals [SKW85, SKC89]; surface implicitization [PP87, PP90, SAG84]; and vector fields [Che88, Kri90].

The loop destruction algorithm requires a method for detecting the possible existence of a loop which in turn makes use of surface-bounding pyramids as reviewed in Section 2. Section 3 then discusses how these pyramids are used to determine the non-existence of closed loops. Section 4 reviews Bézier clipping and applies it to the loop destruction problem.

2 Bounding Pyramids

The loop destruction algorithm makes use of pyramids that bound patches and pyramids that bound tangent directions in a manner discussed below. Several variations on the idea of bounding pyramids have appeared in the literature [SM88, Kim90, Hoh91, Kri90, KP91]. This section reviews, without proof, the bounding pyramids presented in [SZ96].

2.1 Tangent Bounding Pyramids

Given a rational Bézier surface patch $\mathbf{Q}(u,v)$ with positive weights, a *tangent bounding pyramid* $P_u(\mathbf{Q})$ (or $P_v(\mathbf{Q})$) is any pyramid that contains all tangent directions $\mathbf{Q}_u(u,v)$ (or $\mathbf{Q}_v(u,v)$) $0 \leq u,v \leq 1$. This means that any tangent vector whose tail is translated to the appropriate pyramid's vertex will lie within that pyramid.

Denote by $P_{u+}(\mathbf{Q})$ the nappe of $P_u(\mathbf{Q})$ that encloses all the positive tangent directions and denote the other nappe of $P_u(\mathbf{Q})$ by $P_{u-}(\mathbf{Q})$ (see Figure 2.a).

$$\left.\begin{array}{l} P_{u+}(\mathbf{Q}) \supseteq \{\mathbf{Q}_u(u,v) | (u,v) \in [0,1] \times [0,1]\} \\ P_{u-}(\mathbf{Q}) \supseteq \{-\mathbf{Q}_u(u,v) | (u,v) \in [0,1] \times [0,1]\} \end{array}\right\}. \tag{1}$$

Nappes $P_{v+}(\mathbf{Q})$ and $P_{v-}(\mathbf{Q})$ are defined similarly (see Figure 2.b).

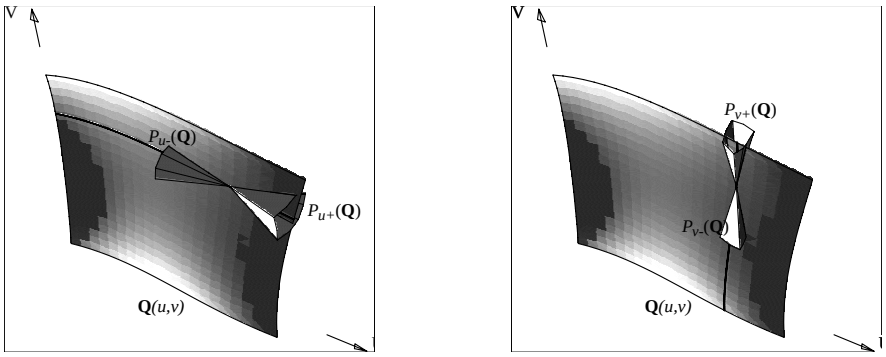

Figure 2: Bézier Surface Patch with U- and V-Tangent Bounding pyramids

For a polynomial tensor product Bézier surface patch

$$\mathbf{Q}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{Q}_{i,j} B_i^m(u) B_j^n(v), \tag{2}$$

the partial derivative is itself a polynomial Bézier patch given by

$$\mathbf{Q}_u(u,v) = \sum_{i=0}^{m-1}\sum_{j=0}^{n}(\mathbf{Q}_{i+1,j} - \mathbf{Q}_{i,j})mB_i^{m-1}(u)B_j^n(v). \tag{3}$$

Because of the convex hull property of Bézier surfaces, the convex hull of the vectors $(\mathbf{Q}_{i+1,j} - \mathbf{Q}_{i,j})$ will contain all partial derivative directions $\mathbf{Q}_u(u,v)$. In other words, the convex hull of the $(\mathbf{Q}_{i+1,j} - \mathbf{Q}_{i,j})$ vectors can serve as $P_{u+}(\mathbf{Q})$. The negative nappe $P_{u-}(\mathbf{Q})$ is similarly the convex hull of the $(\mathbf{Q}_{i,j} - \mathbf{Q}_{i+1,j})$ vectors. $P_v(\mathbf{Q})$ can be constructed in like manner.

For the case of a rational Bézier surface patch, the computation of the hodograph is much more expensive, resulting in a degree $2m \times 2n$ rational patch. A tangent bounding pyramid can be computed as having its vertex at the origin and enclosing all $(2m+1) \times (2n+1)$ hodograph control points. However, [SWS95] presents a much more efficient way of computing the pyramid for a rational patch that only involves vectors defined by control points of the given patch, not of the hodograph.

A useful property of tangent bounding pyramids is that if pyramid $P_u(\mathbf{Q})$ is translated so that its vertex lies at $\mathbf{Q}(\sigma,\tau)$, then the isoparameter curve segment $\mathbf{Q}(u,\tau)$, $u \in [\sigma,1]$, lies within $P_{u+}(\mathbf{Q})$ and $\mathbf{Q}(u,\tau)$, $u \in [0,\sigma]$, lies within $P_{u-}(\mathbf{Q})$ (see Figure 2.a).

In practice, we recommend the use of a pyramid with a rectangular directrix (see [SZ96]).

2.2 Surface Bounding Pyramids

A *surface bounding* pyramid has the property that when its vertex is translated to any point on the surface patch, the surface patch lies entirely outside of the pyramid. Such a pyramid can be derived from two tangent bounding pyramids $P_u(\mathbf{Q})$ and $P_v(\mathbf{Q})$ as follows.

Consider an arbitrary ray in the patch parameter space

$$(u_0 + \alpha t, v_0 + \beta t) \qquad \text{where } \; u_0,\; v_0,\; u_0 + \alpha t,\; v_0 + \beta t \in [0,1]. \tag{4}$$

The image of one of these rays on the surface patch is a parametric curve

$$\mathbf{Q}(u_0 + \alpha t, v_0 + \beta t) \tag{5}$$

whose first derivative (tangent) vector is:

$$\mathbf{Q}_t(u_0 + \alpha t, v_0 + \beta t) = \alpha \mathbf{Q}_u(u_0 + \alpha t, v_0 + \beta t) + \beta \mathbf{Q}_v(u_0 + \alpha t, v_0 + \beta t). \tag{6}$$

Taking the case $\alpha,\ \beta > 0$, we define a pyramid $P_{++}(\mathbf{Q})$ that satisfies

$$\mathbf{Q}_t(u_0 + \alpha t, v_0 + \beta t) \subseteq P_{++}(\mathbf{Q}) = \{\alpha \mathbf{V}_u + \beta \mathbf{V}_v | \mathbf{V}_u \in P_{u+}(\mathbf{Q}),\ \mathbf{V}_v \in P_{v+}(\mathbf{Q})\}. \tag{7}$$

$P_{++}(\mathbf{Q})$ can be taken as the convex hull of $P_{u+}(\mathbf{Q})$ and $P_{v+}(\mathbf{Q})$.

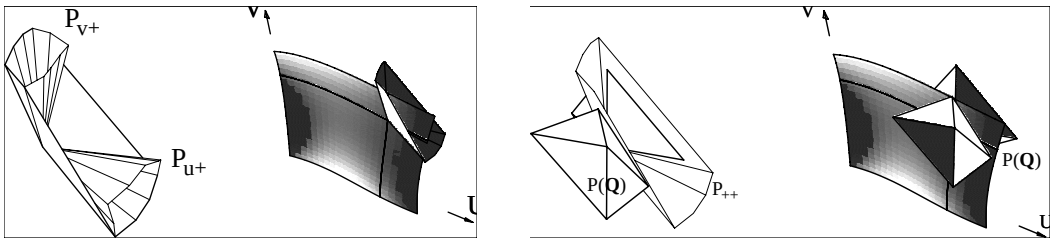

Figure 3: a. P_{++} bounding positive quadrant b. $P(\mathbf{Q})$ surface bounding pyramid

It is shown in [SZ96] that if $P_{++}(\mathbf{Q})$ is translated so that its vertex lies at any point on the surface patch $\mathbf{Q}(u_0, v_0)$, then the portion of $\mathbf{Q}(u, v)$ for which $u \in [u_0, 1]$ and $v \in [v_0, 1]$ will lie entirely within $P_{++}(\mathbf{Q})$.

Likewise we can define bounding volumes $P_{+-}(\mathbf{Q})$, $P_{--}(\mathbf{Q})$ and $P_{-+}(\mathbf{Q})$, each bounding one quadrant of $\mathbf{Q}$. Observe that $P_{+-}(\mathbf{Q})$ and $P_{-+}(\mathbf{Q})$ are opposite nappes of the same pyramid, and $P_{++}(\mathbf{Q})$ and $P_{--}(\mathbf{Q})$ are opposite nappes of the same pyramid. The union of the volumes

$$P_{++}(\mathbf{Q}) \cup P_{+-}(\mathbf{Q}) \cup P_{--}(\mathbf{Q}) \cup P_{-+}(\mathbf{Q}) \tag{8}$$

is the boolean negative of a pyramid, which we denote $P(\mathbf{Q})$:

$$P(\mathbf{Q}) = \{\mathbf{P} = (x, y, z) | \mathbf{P} \notin P_{++}(\mathbf{Q}) \text{ and } \mathbf{P} \notin P_{-+}(\mathbf{Q}) \text{ and } \mathbf{P} \notin P_{--}(\mathbf{Q}) \text{ and } \mathbf{P} \notin P_{+-}(\mathbf{Q})\} \tag{9}$$

$P(\mathbf{Q})$ bounds patch $\mathbf{Q}(u, v)$ in the sense that if the vertex of $P(\mathbf{Q})$ is translated to any point on $\mathbf{Q}(u, v)$, the patch $\mathbf{Q}(u, v)$ will lie entirely outside of pyramid $P(\mathbf{Q})$, as illustrated in Figure 3.b. Note that pyramid $P(\mathbf{Q})$ always has a quadrilateral directrix.

It is shown in [SZ96] that a normal-bounding pyramid (i.e., a pyramid that bounds all surface-normal vectors) can be obtained as the complement of a surface-bounding pyramid

— the locus of all vectors that are not perpendicular to any vector inside the surface-bounding pyramid. Thus, this normal-bounding pyramid has a quadrilateral directrix, and each of its four faces is perpendicular to a face on the surface-bounding pyramid.

3 Loop Detection

Bounding pyramids and cones can be used in several ways to assure the non-existence of a closed loop. When we say that a pair of surfaces passes a loop test, we will mean that the test has positively determined that no loop exists. Test failure is inconclusive; the intersection may or may not involve a closed loop.

> **Loop Test 0** [SKC89]. If either $P_u(\mathbf{Q}_1)$ or $P_v(\mathbf{Q}_1)$ lies completely within $P(\mathbf{Q}_2)$, no closed loop exists.

Figure 4.a shows a directional curve bounding pyramid and a surface bounding pyramid that pass the test, assuring the non-existence of any closed loops. Figure 4.b illustrates a failure of the test indicating a possible loop.

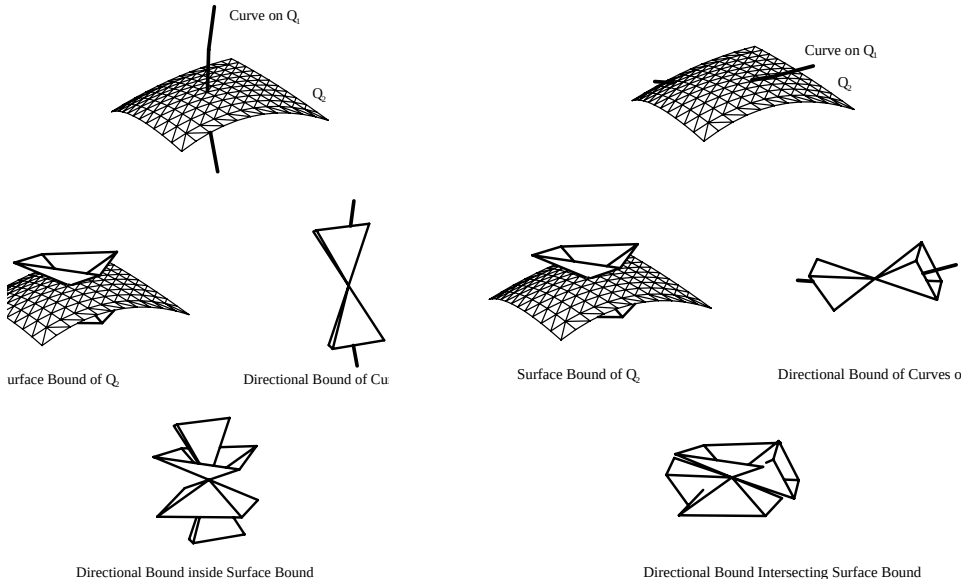

Figure 4: A. Passing the Loop. B. Failing the Loop Test.

Loop Test 0 involves a pyramid that bounds all isoparameter curves of constant u or v. This test can be generalized to consider families of curves whose preimages are parallel straight lines.

$$\mathbf{C}(t, u_0, v_0, \alpha) = \mathbf{Q}(u_0 + \alpha t, v_0 + (1 - |\alpha|t)), \quad u_0, v_0 \in [0, 1], \quad \alpha \in [-1, 1] \tag{10}$$

defines such a family of curves for a fixed α. From the chain rule, the tangent vector for such a curve is

$$\mathbf{C}_t(t, u_0, v_0, \alpha) = \alpha \mathbf{Q}_u(u_0, v_0) + (1 - |\alpha|)\mathbf{Q}_v(u_0, v_0). \tag{11}$$

If $\mathbf{Q}(u, v)$ is a tensor product polynomial surface of degree $m \times n$, $\mathbf{C}(t, u_0, v_0, \alpha)$ is a curve of degree $n + m$ in t. However, for fixed α and for any $u_0, v_0 \in [0, 1]$, $\mathbf{C}_t(t, u_0, v_0, \alpha)$ can be computed by evaluating a degree $n \times n$ hodograph

$$\mathbf{Q}_\alpha(u, v) = \alpha \hat{\mathbf{Q}}_u(u, v) + (1 - |\alpha|)\hat{\mathbf{Q}}_v(u, v) \tag{12}$$

where $\hat{\mathbf{Q}}_u(u, v)$ denotes $\mathbf{Q}_u(u, v)$ after degree elevation in u and $\hat{\mathbf{Q}}_v(u, v)$ denotes $\mathbf{Q}_v(u, v)$ after degree elevation in v. As in Section 2.1, we can determine a pyramid $P_\alpha(\mathbf{Q})$ that bounds the hodograph $\mathbf{Q}_\alpha(u, v)$.

Note that Loop test 0 is actually four tests: If either $P_u(\mathbf{Q}_1)$ or $P_v(\mathbf{Q}_1)$ lies completely within $P(\mathbf{Q}_2)$ — or if either $P_u(\mathbf{Q}_2)$ or $P_v(\mathbf{Q}_2)$ lies completely within $P(\mathbf{Q}_1)$ — no closed loop exists. Figure 5 shows a pair of intersecting surfaces which do not intersect in a closed loop yet all four test 1 cases fail. This is not unusual. However, we can modify the test so that surfaces like this are more likely to pass the test.

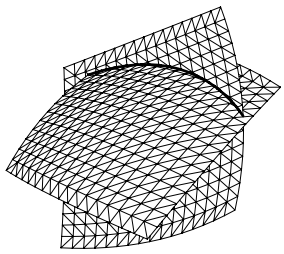

Figure 5: Example Surface Pair

Loop Test 1. If any pyramid $P_\alpha(\mathbf{Q}_1)$ lies completely within $P(\mathbf{Q}_2)$, no closed loop exists.

This follows directly from the proof of Loop Test 0 [SKC89]. In practice, this test is performed several times using various discrete values of α. If the patches pass the test for any value of α, no loop exists.

Loop Test 2 If no vector contained in a surface normal bound is perpendicular to any vector contained in a directional bound (for a fixed parameter direction) on the other surface, no closed loop exists.

This loop test was suggested to the authors by Tim Strotman and is based on the following reasoning. If a closed loop exists, its preimage $\mathbf{C}_1$ on patch 1 is also a closed loop. Any line in the parameter space of patch 1 is parallel to a line $\mathbf{T}$ that is tangent to $\mathbf{C}_i$. At the point of tangency, the image of $\mathbf{T}$ is perpendicular to the normal vector of patch 2 at that point. In practice, Loop Test 2 is equivalent to Loop Test 1, except it uses the normal bound rather than the surface bound.

Test 2 is performed in the following manner.

1. *For each of the four corners (d_i) of the directional bounding pyramid*

   For each of the four corners (n_j) of the normal bounding pyramid
   If $d_i \cdot n_j > max$ set $max = d_i \cdot n_j$.
   If $d_i \cdot n_j < min$ set $min = d_i \cdot n_j$.

2. *If $0.0 \notin [min, max]$ no tangential line exists.*

4 Bézier Clipping

Bézier clipping is the name given to a series of algorithms for performing various computations on Bézier curves and surfaces, such as parametric curve intersection [SWZ89, SN90], curve-surface intersection [SN91], and ray-patch intersection [NSK90]. Here it is used to guide the loop destruction process.

The basic idea behind Bézier clipping is to represent a Bernstein-form polynomial

$$y = f(x) = \sum_{i=0}^{n} y_i B_i^n(x) \tag{13}$$

as an explicit Bézier curve

$$x = t = \sum_{i=0}^{n} \frac{i}{n} B_i^n(t); \quad y = \sum_{i=0}^{n} y_i B_i^n(t). \tag{14}$$

The crucial feature is that the control points are evenly spaced in x along the $[0,1]$ interval. The fundamental operation used in Bézier clipping is to determine ranges of t for which

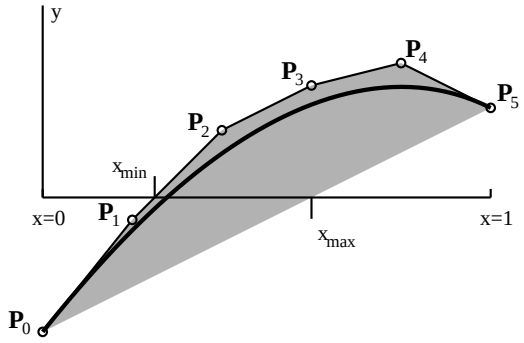

Figure 6: Bézier Clipping

$y(t) \neq 0$. Referring to Figure 6, such ranges can be determined by intersecting the convex hull of the control points of the explicit Bézier curve with the x-axis. Hence, in this example, the ranges $t \in [0, x_{min}]$ and $t \in [x_{max}, 1]$ do not contain a root of $f(t)$.

We now apply this idea to loop destruction. Denote by

$$\mathbf{H}^u(u,v) = \sum_{k=0}^{m} \sum_{l=0}^{n} \mathbf{H}_{kl}^u B_k^m(u) B_l^n(v) \tag{15}$$

the hodograph $\frac{d\mathbf{Q}_1(u,v)}{du}$. $\mathbf{Q}_1$ is a degree $m \times n$ patch, and $\mathbf{H}^u(u,v)$ is degree elevated so that it is also degree $m \times n$.

Consider next a pyramid (with quadrilateral directrix) that bounds the normal vectors of $\mathbf{Q}_2$. Denote by $\mathbf{n}_{00}$, $\mathbf{n}_{01}$, $\mathbf{n}_{10}$, $\mathbf{n}_{11}$ the unit vectors along each edge of the positive nappe of the pyramid and define

$$\mathbf{n}(a,b) = (1-a)(1-b)\mathbf{n}_{00} + a(1-b)\mathbf{n}_{01} + b(1-a)\mathbf{n}_{10} + ab\mathbf{n}_{11}. \tag{16}$$

Then, for any vector normal to $\mathbf{Q}_2$, there exist $a, b \in [0,1]$ such that $\mathbf{n}(a,b)$ is parallel to that normal vector.

Denote

$$D^u(a,b,u,v) = \mathbf{H}^u(u,v) \cdot \mathbf{n}(a,b) = \sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{k=0}^{m} \sum_{l=0}^{n} D_{ijkl}^u B_i^1(a) B_j^1(b) B_k^m(u) B_l^n(v) \tag{17}$$

where

$$D_{ijkl}^u = \mathbf{n}_{ij} \cdot \mathbf{H}_{kl}^u. \tag{18}$$

From Loop Test 2, we observe that if

$$D^u(a,b,u,v) \neq 0, \forall a,b,u,v \in [0,1] \tag{19}$$

there is no closed loop.

We can now adopt the idea of Bézier clipping to identify values of v for which a closed loop cannot exist. Treat $D^u(a,b,u,v)$ as a degree m polynomial in u (in Bernstein form) whose coefficients are polynomials (also in Bernstein form) in the variables a, b, and v:

$$D^u(a,b,u,v) = \sum_{k=0}^{m} \left[\sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{l=0}^{n} D^u_{ijkl} B^1_i(a) B^1_j(b) B^n_l(v) \right] B^m_k(u) \tag{20}$$

Our goal is to identify ranges of v for which $D^u(a,b,u,v)$ is *never* zero for all possible values of $a, b, u \in [0,1]$. This is carried out by expressing $D^u(a,b,u,v)$ as a degree n polynomial in Bernstein form, with interval control points:

$$[D]^u(v) = \sum_{l=0}^{n} [D]_l B^n_l(v). \tag{21}$$

Each of the $[D]_l$ are determined by taking the minimum and maximum of the $4(m+1)$ values

$$D^u_{ijkl}, \quad i = 0,1; \quad j = 0,1; \quad k = 0,\ldots,m. \tag{22}$$

Referring to Figure 7, the gray region shows the projection of $D^u(a,b,u,v)$ onto the v, D^u

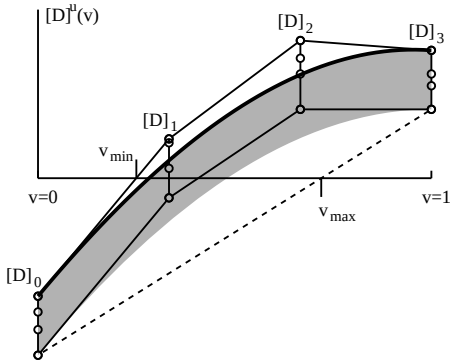

Figure 7: Bézier clipping to determine loop-free regions

plane. For any fixed value of $v = \tilde{v} \in [0,1]$, all values of $D^u(a,b,u,v)$ with $a, b, u \in [0,1]$ lie in the intersection of the gray region with the line $v = \tilde{v}$. Then, since

$$[D]^u(v) \supset D^u(a,b,u,v), \quad a,b,u \in [0,1], \tag{23}$$

it is clear that no loop can exist for any value of v for which

$$[D]^u(v) \not\supset 0. \tag{24}$$

This is easily done by finding where the v axis lies outside of the convex hull of the control points of $[D]^u(v)$. In Figure 7, no loop occurs for $v < v_{min}$ or for $v > v_{max}$. Clearly, if all control points of $[D]^u(v)$ have the same sign, no loops exist anywhere on the patch.

In similar manner, we can determine ranges of u within which no closed loop can occur. This is done by forming an interval Bernstein polynomial $[D]^u(u)$ whose control points $[D]_k$ are determined by taking the minimum and maximum of the $4(n+1)$ values

$$D^u_{ijkl}, \quad i = 0, 1; \quad j = 0, 1; \quad l = 0, \ldots, n. \tag{25}$$

Loop destruction is then attempted by subdividing the patches. If $(u_{max} - u_{min}) < (v_{max} - u_{min})$, then the patch is split in the u direction at $(u_{min} + u_{max})/2$. Otherwise, the split is made in the v direction at $(v_{min} + v_{max})/2$. After splitting, the loop test is again invoked. If the test passes, we're done. Otherwise, further Bézier-clip-guided subdivisions (and subsequent loop tests) are performed.

4.1 Directional Curve Bounding

We have just used the u-hodograph of $\mathbf{Q}_1$ to identify loop-free regions — regions of u and v on $\mathbf{Q}_1$ that cannot contain any closed loops. Likewise, we can find loop-free regions using the v-hodograph of $\mathbf{Q}_1$. Clearly, the union of the loop-free regions obtained used the u- and v-hodographs is also a (possibly larger) loop-free region.

Further, from Loop Test 2, we can potentially enlarge the loop-free regions further by considering hodographs in other parameter directions. Equation 10 defines a curve $\mathbf{C}(t, u_0, v_0, \alpha)$ that is the image of a straight line in parameter space, with arbitrary direction α. We now consider how to determine loop-free regions using $-1 \leq \alpha \leq 1$ — that is, regions where no curve $\mathbf{C}(t, u_0, v_0, \alpha)$ for fixed α is tangent to the SSI curve.

Let

$$\mathbf{H}^\alpha(u, v) = \sum_{k=0}^{m} \sum_{l=0}^{n} \mathbf{H}^\alpha_{kl} B^m_k(u) B^n_l(v) \tag{26}$$

where

$$H^\alpha_{kl} = \alpha H^u_{kl} + (1.0 - |\alpha|) H^v_{kl}. \tag{27}$$

This leads to the function

$$D(\alpha, a, b, u, v) = \mathbf{H}^\alpha(u, v) \cdot \mathbf{n}(a, b) = \sum i = 0^1 \sum j = 0^1 \sum_{k=0}^{m} \sum_{l=0}^{n} D^\alpha_{ijkl} B^1_i(a) B^1_j(b) B^m_k(u) B^n_l(v) \tag{28}$$

where

$$D^\alpha_{ijkl} = \mathbf{n}_{ij} \cdot \mathbf{H}^\alpha_{kl} = \mathbf{n}_{ij} \cdot (\alpha \mathbf{H}^u_{kl} + (1.0 - |\alpha|) \mathbf{H}^v_{kl}). \tag{29}$$

For any given α, loop free regions may be determined by performing a Bézier clip on the functions $[D]^\alpha(u)$ and $[D]^\alpha(v)$.

Since Bézier clip calculations are relatively cheap, it generally pays to check a few values of α in addition to 0 and 1.

4.2 Spatial Bézier Clipping

It often happens that the bounding boxes of two patches often overlap only slightly. Figure 8 shows two surfaces for which the loop detection tests fail. Since the tests are translation

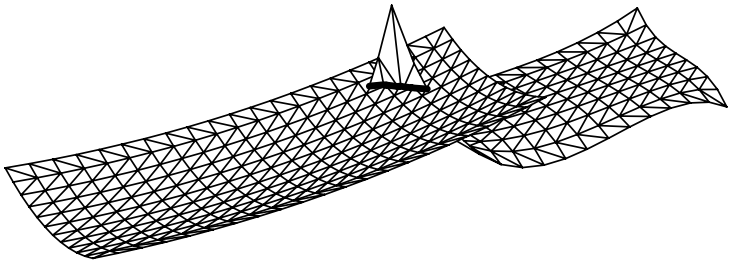

Figure 8: Surface Pair with Small Overlap

invariant, if the patches can be translated so as to create a closed loop, the test will fail. In this example, the intersection is transverse, but the test fails since the patches can be translated so as to create a closed loop.

An effective solution to this problem, suggested by Tomoyuki Nishita, is to perform a Bézier clip of each patch against the bounding box of the other patch. This operation clips away portions of one patch which do not lie within the bounding box of the opposing patch, resulting in two smaller patches that tend to more nearly occupy roughly the same region in space. Thus, regions of each patch which definitely play no roll in the intersection have no influence on the loop detection test. The results of such a clip on the surface patch pair shown in Figure 8 are shown in Figure 9. In this refined problem, the loop test passes.

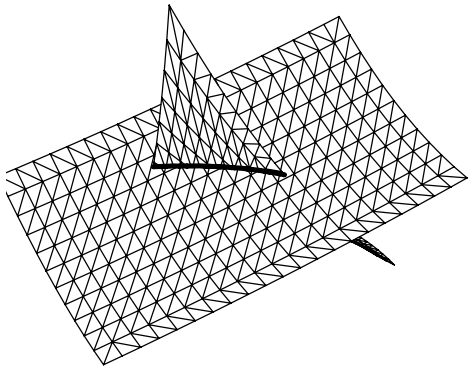

Figure 9: Surface Pair after Bézier Clipping to Bounding boxes

5 Discussion

The power of Bézier clipping for loop destruction is dramatically illustrated in the case of a simple convex surface just tangent to a plane (Figure 10), where there exists an infinitesimally small closed loop. In this contrived case, all rows of the control grid are translations of each other, so all the rows of the u-hodograph are identical. Therefore, $[D]^u(u)$ becomes an interval function of width zero. Further, since the surface is convex, the function is monotonic with one root. This root locates the u value of the tangency point between the surface and the plane exactly. The v location can be computed in a similar manner using

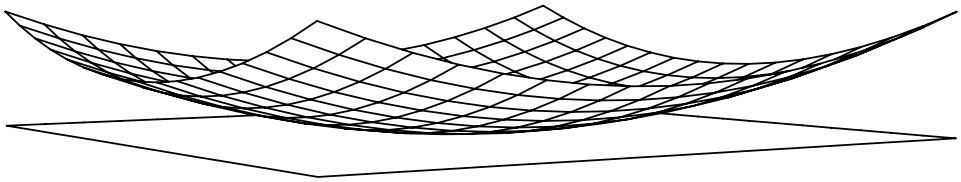

Figure 10: Simple Convex Surface Tangent to Plane

the v hodograph and $[D]^v(v)$.

The example in Figure 11 involves two fairly complicated surfaces which intersect in four branches, three of which are closed loops. Loop detection obviously fails, requiring

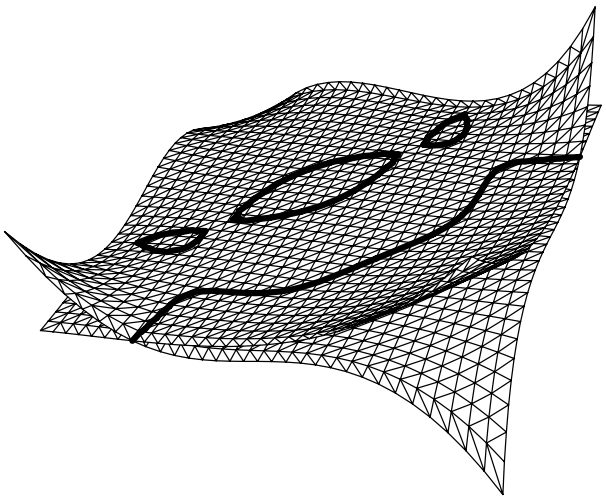

Figure 11: Surface Intersection Problem With Multiple Loops

subdivision. The problem is so complicated that Bézier clipping fails to identify any loop-free region, so a bisection is performed. One of the loops is split into open branches, and the remaining two loops are separated into subproblems. One of these subproblems is illustrated in Figure 12. Both of the subproblems also fail the loop detection tests. On the third iteration, Bézier clipping identifies loop-free regions, guiding the subdivision to sucessfully split the two remaining loops.

References

[Arn87] P. R. Arner. *Another Look at Surface/Surface Intersection.* Deptartment of computer science, University of Utah, 1987.

[BHHL88] C. L. Bajaj, C. M. Hoffmann, J. E. Hopcroft, and R. E. Lynch. Tracing surface intersections. *Computer Aided Geometric Design*, 5:285–307, 1988.

[BK90] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface/surface intersections. *Computer Aided Geometric Design*, 7:257–280, 1990.

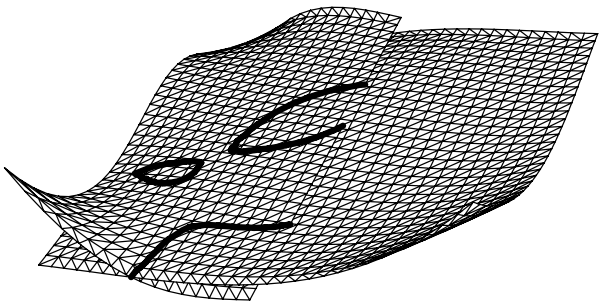

Figure 12: Multiple Loop Subproblem

[Che88] K. P. Cheng. Using plane vector fields to obtain all the intersections of two general surfaces. In W. Strasser, editor, *Theory and Practice of Geometric Modeling*, pages 187–204. Springer-Verlag, 1988.

[CO88] J. J. Chen and T. M. Ozsoy. Predictor-corrector type of intersection algorithm for c^2 parametric surfaces. *Computer Aided Design*, 20(6):347–352, 1988.

[Dok85] T. Dokken. Finding intersections of b-spline represented geometries using recursive subdivision techniques. *Computer Aided Geometric Design*, 2:189–195, 1985.

[HAG83] S. L. Hanna, J. F. Abel, and D. P. Greenberg. Intersection of parametric surfaces by means of lookup tables. *IEEE Computer Graphics and Applications*, 3(7):39–48, 1983.

[HEFS85] E. G. Houghton, F. F. Emnett, J. D. Factor, and C. L. Sabharwal. Implementation of a divide-and-conquer method for intersection of parametric surfaces. *Computer Aided Geometric Design*, 2:173–183, 1985.

[Hoh91] Michael Hohmeyer. A surface intersection algorithm based on loop detection. *International Journal of Computational Geometry and Applications*, 1(4):473–490, 1991.

[Kim90] Deok-Soo Kim. *Cones on Bézier curves and surfaces.* PhD thesis, University of Michigan, 1990.

[KP91] G. A. Kriezis and N. M. Patrikalakis. Rational polynomial surface intersections. In G. A. Gabriele, editor, *Proceedings of the 17th ASME Design Automation Conference*, volume II, pages 43–53, 1991.

[Kri90] George A. Kriezis. *Algorithms for Rational Spline Surface Intersections.* PhD thesis, Massachusetts Institute of Technology, 1990.

[KS88] Sheldon Katz and Thomas W. Sederberg. Genus of the intersection curve of two rational surface patches. *Computer Aided Geometric Design*, 5(3):253–258, September 1988.

[LR80] J. M. Lane and R. F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1), 1980.

[NSK90] T. Nishita, T. W. Sederberg, and M. Kakimoto. Ray tracing trimmed rational surface patches. *Computer Graphics*, 24:337–345, 1990.

[PP87] N. M. Patrikalakis and P. V. Prakash. Computation of algebraic polynomial parametric surface intersections. Technical report, MITSG 87-19, MIT Sea Grant College Program, 1987.

[PP90] N. M. Patrikalakis and P. V. Prakash. Surface intersections for geometric modeling. *Journal of Mechanical Design, ASME Transactions*, 1990.

[Pra86] M. J. Pratt. Surface/surface intersection problems. In J. A. Gregory, editor, *The Mathematics of Surfaces II*. Oxfor, Claredon Press, 1986.

[SAG84] Thomas W. Sederberg, D. C. Anderson, and Ron N. Goldman. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics and Image Processing*, 28:72–84, 1984.

[SKC89] Thomas W. Sederberg, Sheldon Katz, and Henry N. Christiansen. An improved test for closed loops in surface intersections. *Computer Aided Design*, 21(8):505–508, October 1989.

[SKW85] P. Sinha, E. Klassen, and K. K. Wang. Exploiting topological and geometric properties for selective subdivision. In *Proc. ACM Symposium on Computational Geometry*, All ACM Conferences, pages 39–45, 1985.

[SM88] Thomas W. Sederberg and Ray J. Meyers. Loop detection in surface patch intersections. *Computer Aided Geometric Design*, 5:161–171, 1988.

[SN90] Thomas W. Sederberg and Tomoyuki Nishita. Curve intersection using bezier clipping. *Computer-Aided Design*, pages 538–549, 1990.

[SN91] Thomas W. Sederberg and Tomoyuki Nishita. Geometric hermite approximation of surface patch intersections curves. *Computer Aided Geometric Design*, 8:97–114, 1991.

[SWS95] Takafumi Saito, Guo-Jin Wang, and Thomas W. Sederberg. Hodographs and normals of rational curves and surfaces. *Computer Aided Geometric Design*, 12:417–430, 1995.

[SWZ89] Thomas W. Sederberg, Scott C. White, and Alan K. Zundel. Fat arcs: A bounding region with cubic convergence. *Computer Aided Design*, 6:205–218, 1989.

[SZ96] Thomas W. Sederberg and Alan K. Zundel. Pyramids that bound surface patches. *CVGIP: Graphical Models and Image Processing*, 58:75–81, 1996.

[Tim77] H. G. Timmer. A solution to the surface intersection problem. Technical report, McDonnel Douglas Report No. MDC J7789, 1977.