Surface-to-surface intersection based on triangular parameter domain subdivision

Ernst Huber*
Institute of Computer Graphics
Vienna University of Technology
A-1040 Vienna, Karlsplatz 13/186/1, Austria.

Abstract: *An improved algorithm for the computation of the intersection curve of two general parametric surfaces is presented. The introduced subdivision algorithm follows a divide-and-conquer-approach. For each pair of patches, it first checks if the corresponding bounding volumes intersect. If they intersect, then it tries to find dispensable parts of one patch (and the corresponding parts in the parameter domain) which can be cut off. Next it splits the reduced patch by splitting its parameter domain into triangular pieces and treats all new pairs of patches recursively until a predefined termination condition is satisfied. To take full advantage of the cut-off operation patches with triangular domains bounded by so-called "Tripipeds" are used.*

1 Introduction

This paper introduces an algorithm for computing an approximative solution for the general surface-to-surface intersection (*SSI*) problem. *SSI* is a basic problem in Computer Aided Geometric Design, it can be stated very simply: *Given are two intersecting surfaces in $\mathbb{R}^3$, compute all parts of the intersection curve.*

If two surfaces intersect, the result will be a set of isolated points, a set of curves, a set of overlapping surfaces, or any combination of these cases [1]. Because exact solutions can be found only for some special surface classes, approximation methods must be used for the general case.

The basic algorithm was introduced in [3]. It is a robust subdivision algorithm which computes an enclosure for all parts of the intersection curve of two general parametric surfaces. Surfaces are defined as

$$\vec{s}(u,v) = \left\{ \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} \;\middle|\; (u,v) \in \mathcal{D}_\square \right\}, \qquad (1)$$

where $\mathcal{D}_\square = U \times V$, and U, V are two intervals[1] in $\mathbb{R}$. The coordinate functions map a rectangular parameter domain to object space. It is only assumed that these functions are continuous in the whole domain and that the partial derivatives exist and are continuous.

The basic algorithm follows a divide-and-conquer approach. For each pair of patches, it first checks for intersection of the corresponding bounding volumes. If two bounding volumes intersect, it splits one patch into two subpatches and treats both new pairs recursively until a predefined termination condition is satisfied. The shape of the parameter domain of a patch determines in a certain way the shape of the corresponding bounding volume. The basic algorithm divides a patch by dividing its parameter domain vertically or horizontally, the resulting domains are always axis aligned. In object space we get patches with four corner points which can be enclosed by a tight parallelepiped in considering the shape and orientation of the patch.

This work introduces an improved algorithm working with triangular domains. Triangular domains offer more freedom than axis aligned domains, because they allow us to subdivide a patch in an arbitrary manner by triangulating its domain. In object space we get patches with three corner points which can be approximated by triangles and enclosed by a bounding volume we call "Tripiped". This allows to take full advantage of an optimization step which detects parts of a patch (and the corresponding parts in the parameter domain), where one patch cannot reach the enclosure of the other one. Cutting off such dispensable parts leads to smaller domains, smaller bounding volumes and fewer subdivisions, thus speeding up the overall algorithm.

```
// Improved divide-and-conquer SSI-algorithm
procedure intersect(Surface s,t)
  if boundVolumesIntersect(s,t) then        // Section 2,3
    if termCondition(s,t) then
      intersectBoundingVolumes(s,t)
    else
      s* ← cutDomain(s, boundVolume(t))     // Section 4
      splitSurface(s*, s1, s2, ..., sn)     // Section 4
      for i ← 1 to n do intersect(t, si)
```

Similar to the basic algorithm the improved algorithm

*huber@apm.tuwien.ac.at.

[1] Notation: small letters denote scalar values and functions; capital letters denote intervals and functions returning intervals with $A = [\underline{a}, \overline{a}] = \{a \in \mathbb{R} : \underline{a} \le a \le \overline{a}\}$; vectors are denoted by $\vec{a}$.

first checks for intersection of the corresponding bounding volumes. The difference is in the following steps: If two bounding volumes intersect, it tries to find dispensable parts of the domain of patch $\vec{s}$ which can be cut off. The reduced patch $\vec{s}^*$ is split into two or *more* subpatches $\vec{s}_i$ with triangular domain. Then all new pairs of patches are treated recursively until a predefined termination condition is satisfied.

The result is a collection of intersecting bounding volumes in object space and a collection of pieces for both parameter domains. Both representations are enclosures for all parts of the corresponding intersection curve. The usage of bounding volumes guarantees that all intersection points are detected.

Crucial steps are the subdivision of the surfaces, the computation of tight bounding volumes and the test whether two bounding volumes intersect.

2 Enclosing triangular surface-parts

We start with the construction of an enclosure for a patch $\vec{s}$ of the original surface. The patch is defined as

$$\vec{s}(u,v) = \left\{ \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} \middle| \quad (u,v) \in \mathcal{D}_\triangle \right\}, \qquad (2)$$

where $\mathcal{D}_\triangle$ is the triangular domain defined by three corner points (u_j, v_j), $j \in \{1,2,3\}$. Since the original surface is defined for a rectangular domain, we may further define $\mathcal{D}_\square = U \times V$ as the smallest axis aligned bounding rectangle for $\mathcal{D}_\triangle$, so that $\mathcal{D}_\triangle \in \mathcal{D}_\square$. By means of interval arithmetic and automatic differentiation we compute inclusion intervals for the partial derivatives with respect to u and v for the coordinate functions, e.g. for $x(u,v)$:

$$\left.\begin{array}{rcl} x_u(u,v) & \in & X_u \\ x_v(u,v) & \in & X_v \end{array}\right\} \forall (u,v) \in \mathcal{D}_\square.$$

For each coordinate function we apply the mean value theorem of differential calculus for two independent variables, e.g. for $x(u,v)$:

$\forall\ u, u_0 \in U, v, v_0 \in V$ exists a $\tau_u \in [u, u_0] \subseteq U, \tau_v \in [v, v_0] \subseteq V$, so that

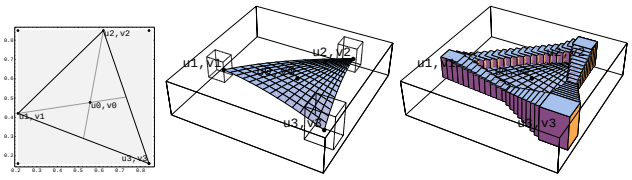
$$x(u,v) = x(u_0,v_0) + (u-u_0)x_u(\tau_u, v_0) + (v-v_0)x_v(u,\tau_v). \qquad (3)$$

Replacing the partial derivatives by the corresponding intervals, u by u_1, and v by v_1 lets us rewrite for the corner point $\vec{c}_1 = \vec{s}(u_1, v_1)$:

$$\vec{c}_1 \in \vec{B}(u_1,v_1) = \vec{s}(u_0,v_0) + (u_1-u_0)\vec{S}_u + (v_1-v_0)\vec{S}_v, \quad (4)$$

where $\vec{S}_u = (X_u, Y_u, Z_u)$, $\vec{S}_v = (X_v, Y_v, Z_v)$, and $(u_0, v_0) \in \mathcal{D}_\triangle$. This equation provides a bounding box $\vec{B}(u_1, v_1)$ containing the corner point $\vec{c}_1$. Repeating this step for the other corner points and showing that the three corner boxes can be transformed to each other by a linear transformation allows to state the corollary:

Corollary 1 *Each convex volume enclosing the three corner boxes $\vec{B}(u_j, v_j)$, $j \in \{1,2,3\}$, is a bounding volume for the patch $\vec{s}(u,v)$.*



(a) (b) (c)

Figure 1: Enclosing triangular patches.

3 Tripipeds

The next step is to construct a tight bounding volume for this enclosure. The three corner points of the patch define the orientation of a plane supporting a triangle, which approximates the patch. We take a pair of such planes (a top and a bottom plane) and move each plane along its normal vector until all bounding boxes $B(u_j, v_j)$ are completely contained in the slab. The orientation of the planes supporting the three other faces is defined by a vector normal to the top and bottom plane and a vector supporting an edge of the triangular approximation of the patch. Each plane is moved along its normal vector until all bounding boxes $B(u_j, v_j)$ lie inside the so-called "Tripiped".

For testing whether two Tripipeds intersect we apply the following theorem [2]:

Theorem 1 *Two convex polyhedra do not intersect if and only if there exists a separating plane which is either parallel to a face of one polyhedron or which is parallel to at least one edge of each polyhedron.*

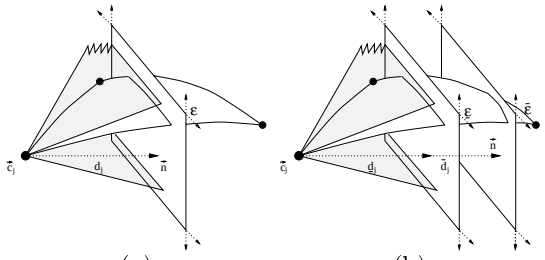
An axis normal to a separating plane is called a *separating axis*. Projecting both polyhedra on a separating axis (by an orthogonal projection) we obtain an interval for each polyhedron.

If at least one separating axis exists, where the projected polyhedra (the intervals) do not intersect, then the polyhedra do not intersect (otherwise the polyhedra intersect).

4 Dispensable surface parts

With the help of interval inclusions for the partial derivatives, we compute parts of the parameter domain, where one patch cannot reach the enclosure of the other one. Cutting off such dispensable regions, and corresponding parts in object space result in a faster convergence of the algorithm.

Let us consider a patch $\vec{s}$ and a plane ε which is given by a normal vector $\vec{n}$ and the distance λ of the plane from the origin. The problem is stated as follows: *Determine parts of $\vec{s}$ and the corresponding regions in the parameter domain which definitely lie on the same side of ε as a corner point $\vec{c}_j$ of $\vec{s}$.*



(a) (b)

Figure 2: Finding dispensable surface parts.

For each point of the patch, we can compute an enclosure by a similar application of the mean value theorem of two independent variables as in Section 2:

$$\vec{s}(u,v) \quad \in \quad \vec{c}_j + \underbrace{(u-u_j)}_{\Delta u}\vec{S}_u + \underbrace{(v-v_j)}_{\Delta v}\vec{S}_v. \qquad (5)$$

We start with the computation of the distance d between a corner point $\vec{c}_j$ and ε. We may state: A surface point $\vec{s}(u,v)$ lies on the same side as $\vec{c}_j$ if all points of its enclosure $(\vec{c}_j + \Delta u\vec{S}_u + \Delta v\vec{S}_v)$ lie on the same side as $\vec{c}_j$. The condition for this is:

$$\begin{aligned} (\Delta u\vec{S}_u + \Delta v\vec{S}_v)\cdot\vec{n} &< d_j, && \text{if } d_j \geq 0,\\ (\Delta u\vec{S}_u + \Delta v\vec{S}_v)\cdot\vec{n} &> d_j, && \text{otherwise.} \end{aligned}$$

Only points $\vec{s}(u,v)$, where the according points (u,v) of the parameter domain solve the linear interval equation

$$(\Delta u\vec{S}_u + \Delta v\vec{S}_v)\cdot\vec{n} \quad = \quad d_j, \qquad (6)$$

can lie on ε. Thus Equation (6) provides boundaries (a polygonal line) separating the parameter domain of the patch into three regions:

1. a region defining all points of $\vec{s}$ which definitely lie on the same side of ε as $\vec{c}_j$,
2. a region defining all points of $\vec{s}$ which possibly lie on ε, and
3. a region defining all points of $\vec{s}$ lying definitely on the other side of ε.

Note that region 2 must contain all points that cannot be unambiguously assigned to region 1 or 3 due to the fact that $\vec{S}_u$ and $\vec{S}_v$ are intervals when computing d and solving Equation (6).

We extend this construction for a pair of planes $(\underline{\varepsilon}, \overline{\varepsilon})$ defining a slab (the top and bottom plane of the bounding Tripiped for the second patch, see Figure 2b).

For each corner point $\vec{c}_j$ of patch $\vec{s}$ we do the following steps: We check if the corner point lies outside the slab. If this true, then we take the plane closer to $\vec{c}_j$ and solve the linear Equation (6). As result we get a (polygonal) line separating the region which can be cut off (shown in Figure 3 for the highest and lowest corner point).

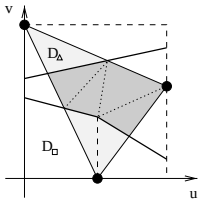


Figure 3: Reduced parameter domain.

5 Results

We are currently implementing the algorithm as part of an experimental system (see *http://www.apm.tuwien.ac.at/research/ssi/*). We constructed algorithms working with patches with rectangular or triangular domains. We use parallelepipeds for patches with rectangular domain, and Tripipeds for patches with triangular domain as tight bounding volumes. Axis aligned bounding boxes can be used for both patch types for a first intersection test. True interval arithmetic is used for all critical operations to achieve a robust algorithm.

First experiments have shown that the cut-off operations increase the speed of the overall algorithm by a factor of 10 by decreasing the number of investigated patches to $\frac{1}{50}$ (results depend strongly on the surfaces, more detailed results are given in the final paper).

Further investigations will be made for finding a proper termination condition (at the moment we only check the size of the parameter domain of a patch), for finding a fast way of computing the intersection volume of two Tripipeds and on data structures for storing the output of the algorithm.

References

[1] R. Barnhill, G. Farin, M. Jordan, and B. Piper. Surface/Surface Intersection. *Computer Aided Geometric Design*, 4:3–16, 1987.

[2] S. Gottschalk. Separating Axis Theorem. Technical Report TR96-024, Department of Computer Science, UNC Chapel Hill, 1996.

[3] E. Huber. Intersecting General Parametric Surfaces Using Bounding Volumes. In *Tenth Canadian Conference on Computational Geometry - CCCG'98*, 1998.