Surface-Surface-Intersection Computation using a Bounding Volume Hierarchy with Osculating Toroidal Patches in the Leaf Nodes

Youngjin Park*, Sang-Hyun Son*, Myung-Soo Kim*, Gershon Elberb

*Dept. of Computer Science and Engineering, Seoul National University, Seoul 08826, South Korea
bComputer Science Department, Technion, Haifa 32000, Israel

Abstract

We present an efficient and robust algorithm for computing the intersection curve of two freeform surfaces using a Bounding Volume Hierarchy (BVH), where the leaf nodes contain osculating toroidal patches. The covering of each surface by a union of tightly fitting toroidal patches greatly simplifies the geometric operations involved in the surface-surface-intersection computation, i.e., the bounding of surface normals, the detection of surface binormals, the point projection from one surface to the other surface, and the intersection of local surface patches. Moreover, the hierarchy of simple bounding volumes (such as rectangle-swept spheres) accelerates the geometric search for the potential pairs of surface patches that may generate some curve segments in the surface-surface-intersection. We demonstrate the effectiveness of our approach by using test examples of intersecting two freeform surfaces, including some highly non-trivial examples with tangential intersections. In particular, we test the intersection of two almost identical surfaces, where one surface is obtained from the same surface, using a rotation around a normal line by a smaller and smaller angle $\theta = 10^{-k}$ degree, $k = 0, \cdots, 5$. The intersection results are often given as surface subpatches in some highly tangential areas, and even as the whole surface itself, when $\theta = 0.00001^\circ$.

Keywords: Surface-surface-intersection, bounding volume hierarchy, rectangle-swept sphere (RSS), osculating toroidal patches

1. Introduction

The problem of intersecting two freeform curves and surfaces in a reliable way is one of the main technical challenges in geometric and solid modeling [16,17,38,41]. Unfortunately, in the last decade, there have been only a few publications in this fundamental research [21,53]. On the other hand, the problem of intersecting two surfaces appears in many different forms as the solution of non-linear geometric constraints [4,7,12], where the constraints are often represented or automatically generated as rational freeform curves, surfaces, and multivariate volumes. Consequently, it is still extremely important to develop new techniques that can handle the surface-surface-intersection (SSI) problem in an efficient and reliable way.

Now, a natural question arises: what makes the revisiting of this problem worthwhile for some meaningful technical advancements? To answer this question in a positive perspective, we would like to mention the recent developments of spatial data structures (such as bounding volume hierarchy) for freeform curves and surfaces, which have made possible the acceleration of many important geometric algorithms, including collision detection, the minimum and Hausdorff distance computations, offset curve trimming, Minkowski sum computation, medial axis and Voronoi diagram construction, convex hull computation, etc. [13,22,24,26].

Planar freeform curves can be approximated by a hierarchy of $G^1$-biarcs with cubic convergence, which can be used for the construction of arc trees (i.e., BVH structures where the internal nodes are fat arcs [24]). The arc trees are a very powerful tool for the acceleration of algorithms for planar freeform curves. A natural dimension-extension of arc trees seems to be torus trees. Nevertheless, different from the planar case, it is extremely difficult to approximate freeform surfaces with $G^1$-continuous toroidal patches. As a compromise, we consider a union of osculating toroidal patches that tightly fits the given freeform surface. (We use the second order osculating torus patch of Liu et al. [32]). Thanks to the higher approximation order than triangular and bilinear approximations, the osculating toroidal patches provide good approximate solutions to the SSI problem even though they are not connected even with $C^0$-continuity.

In this work, we consider the acceleration of the SSI problem using a BVH for freeform surfaces. There are certainly some costs we have to pay for the memory space and the preprocessing time for the BVH construction. Nevertheless, the majority of recent acceleration algorithms are based on spatial data structures (often precomputed) at some additional costs. For example, an alternative approach (to the SSI problem) using triangulated meshes of the freeform surfaces would also require the construction of BVH trees in a preprocessing stage, often taking considerably more time and space than our approach. As the memory cost becomes cheaper, we may even assume the pre-constructed BVH structure as a part of the surface representation by encoding the structure compactly as a 2D array, much like an image, which is the approach we take in the current work.

For the sake of simplicity in the presentation, we consider the intersection of two bicubic Bézier surfaces, though the basic approach can easily be extended to higher degree trimmed B-spline surfaces or even to triangular Bézier surfaces. Taking uniform samples along each parameter direction, $S\left(\frac{i\pi}{2^j}, \frac{j\pi}{2^j}\right)$, where $i, j = 0, \cdots, 2^H$, the BVH structure for a bicubic Bézier surface $S(u,v)$, $(0 \leq u, v \leq 1)$, can be represented in about the same way as that for the quadtree (of height $H$) for regular quad meshes (Figure 1). The main difference is in the internal nodes which contain bounding volumes slightly thicker than those for a regular quad mesh. The extra thickness is needed for bounding the difference between a surface $S(u,v)$ and the quad mesh ap-
proximation (interpreted as a rectangular array of bilinear surface patches or pairs of triangles).

The surface approximation error can be bounded using a simple formula developed by Filip et al. [9]. (Krishnamurthy et al. [25, 26] also use this formula for the construction of various BVH trees for freeform surfaces.) Using a 2D array of size \((1 + 2^h) \times (1 + 2^h)\), where the \(x, y, z\) coordinates of \(S(\frac{x}{2^h}, \frac{y}{2^h})\) are stored in each element, we can encode the BVH structure of the surface \(S(u, v)\) in a compact way, where the thickness of the internal nodes at a level \(h (\leq H)\) can be estimated by the formula of Filip et al. [9]. (This is because the higher level \(h\) approximates the surface \(S(u, v)\) using a coarser quad mesh with vertices taken at \(S(\frac{x}{2^h}, \frac{y}{2^h})\), where \(i, j = 0, \cdots, 2^h\).

The BVH-based intersection algorithm for freeform surfaces is about the same as other conventional algorithms for meshes. The main difference is in the intersection of two primitives stored in the leaf nodes. Instead of intersecting two triangles, we intersect two oscillating toroidal patches, which may produce intersection curve segments of more complex shapes such as those with multiple branches and even X-junctions. The first row of Figure 2 shows two examples of the tangential intersection, each with an X-junction. As we move the blue patch up and down along the normal direction, the X-junction disappears and the intersection curve splits to two branches as shown in the second and third rows of the figure.

When two surfaces intersect tangentially at a point, they share the same normal line (i.e., their binormal line) at the point. The location of an X-junction or a near X-junction can be detected by computing the binormal lines to the two surfaces and checking the signed distance between the corresponding binormal-surface intersection points. Depending on the sign, we can decide which type of (almost) tangential intersection curve the two surfaces have. For example, in Figure 2 when measured along the normal direction of the blue torus, the binormal-surface intersection point of the red torus is located at a signed distance of zero, negative, and positive, respectively, in each of the three rows. Osculating toroidal patches greatly simplify this test by converting the distance computation to a much simpler problem of solving a polynomial equation of degree 8 [36, 44]. The polynomial equation can be solved considerably faster than the general case of computing binormal lines for two bivariate surfaces, where we need to deal with a system of four equations in four variables.

For toroidal patches, their Gauss maps are easy to compute. When there is no overlap in the Gauss maps of the toroidal patches, the intersection curve may have at most one connected branch, which can be traced very efficiently using numerical techniques. Otherwise, we take further steps (such as subdivision or resampling of osculating tori) for detecting the correct topology of SSI curves. Though this is quite a complex task in general due to tangential surface intersections, the binormal construction algorithm for toroidal patches greatly simplifies this process. We subdivide the surfaces at the binormal-surface intersection points, which can remove an X-junction or reduce the number of branches in the intersection curve. In other words, we can make (almost) X-junctions appear only at (or around) the corner points of surface patches, after the subdivision.

The numerical curve tracing can be carried out either on the original surfaces or on the osculating toroidal patches. According to the experimental results reported in Section 5, the difference is insignificant as we cover the entire surface using a union of tightly fitting osculating toroidal patches within a small Hausdorff distance bound, such as \(\epsilon = 10^{-5}\).

The main contributions of this work can be summarized as follows:

- We propose a new BVH-based algorithm for computing the SSI curves for freeform surfaces, which outperforms other conventional algorithms in computing speed and robustness by computing the BVH structure in a preprocessing time.
- The improvement is based on the high approximation order of osculating toroidal patches to the freeform surfaces and the geometric simplicity of toroidal patches in supporting primitive operations (needed for the SSI computation) such as the Gauss map computation, the overlap test between two Gauss maps, the binormal line construction, the projection of points onto a surface, etc.
- The BVH of freeform surfaces can be represented in a compact way using a rectangular array of uniform sampling points and other simple spatial data structures which resemble the mipmap structure for texture mapping.
- The main advantage of our SSI algorithm is in handling the degenerate case of (almost) tangential surface intersections.
where the conventional subdivision methods have difficulty pruning a large number of potentially overlapping pairs of small surface patches.

- The osculating toroidal patches in the leaf nodes not only bound the small surface patches corresponding to the leaf nodes, but they also bound their surface normals. This will be discussed in Section 3.

- Further trimming the toroidal patches into a shape similar to the surface patches (using a surface matching technique to be discussed briefly at the end of Section 3), we can bound their Hausdorff distance within a small tolerance \( \epsilon = 10^{-5} \) (by repeating the local subdivisions recursively if necessary).

- The 1-1 matching between each local surface patch and the corresponding trimmed toroidal patch (within a Hausdorff distance \( \epsilon = 10^{-5} \)) makes the SSI curve constructed in the \( u \)-parameter space of one surface \( S_j(u,v) \) also 1-1 matched with its counterpart SSI curve in the \( \nu \)-space of the other surface \( S_j(s,t) \). (The Hausdorff distance between the two matching SSI curves is then bounded by \( k \epsilon = k \cdot 10^{-5} \), where \( k \) is a small constant.) More details will be discussed in Section 4.

2. Related work

A comprehensive introduction to the problem of intersecting freeform curves and surfaces can be found in the Chapter 12 of Hoschek and Lasser [17], where early methods are explained in great details, including algebraic, subdivision, embedding, discretization, and tracing methods. In a survey article (published in 1993), Patrikalakis [15] reviews the SSI algorithms developed in the period of 1988–1992, which are then classified into four main categories: analytic, lattice evaluation, marching, and subdivision methods. Farin [8] compiled an SSI bibliography of 50 references, containing algorithms developed in the period of 1968–1990. On the other hand, Patrikalakis and Maekawa [16] introduce an extensive body of SSI literature (with more than 100 references), including new results developed in the late 90’s, in particular, some techniques that can deal with the robustness issues using interval arithmetics. Since then, there have been relatively few publications in the SSI research. New results are often those for special types of surfaces such as torus [21, 22], or sweep surfaces [15, 23], or algorithms [18, 19] dealing with special cases that had been overlooked in the previous work.

The GPU-based SSI approach of Krishnamurthy et al. [25] generates the BVH structures for freeform surfaces (partially and dynamically on the fly). Because of the non-flexibility of GPU implementation, Krishnamurthy et al. [25] constructed only AABB (Axis-Aligned Bounding Box) trees; nevertheless, their basic approach can be applied to the generation of other BVH structures for the SSI problem (including CPU-based implementations). The BVH-based approach belongs to the category of the subdivision method. (The subdivisions are often made globally all over the surface in a preprocessing stage of the BVH construction.)

At the end of the BVH traversal for the SSI computation, a tracing method is then needed for the local construction of intersection curve segments. At the curve tracing stage, our approach is different from the conventional BVH-based algorithms for intersecting two mesh models, where the intersection of two triangles typically produces a line segment [1, 27, 30, 31]. (In some degenerate cases, two triangles may overlap in a convex polygon only when they are contained in the same plane.) On the other hand, the intersection of two toroidal patches can be more complex (as shown in Figure 2).

For the resolution of the SSI curve topology in space, Sederberg et al. [40, 41] developed methods for detecting closed loops in the SSI curve by comparing the Gauss maps of the two surface patches to be intersected. When there is no overlap in their Gauss maps, the SSI curve has (at most) a single branch (not forming a closed loop). Otherwise, we can recursively subdivide the surfaces into smaller patches until the Gauss maps have no overlap. Nevertheless, in the degenerate case of (almost) tangential surface intersections, we need a long sequence of recursive subdivisions and at some point we have to deal with the SSI curve with multiple branches and/or closed loops. Even in these non-trivial cases, the osculating toroidal patches provide good approximate solutions to the tangential SSI problem, thanks to their geometric simplicity and high approximation order to the surface.

Regarding the robustness issue, the bounding volumes may be interpreted as a geometric version of the interval arithmetics (in the sense that intervals are one-dimensional AABBs) or a simplified version of other conservative techniques such as the convex hulls of control points for surface patches. The BVH-based SSI approach is thus a generalization of conventional SSI methods, where the main difference is that the subdivisions are done globally in a preprocessing stage, to a certain high resolution such as 512 x 512. The traversal in the hierarchy of surface subdivisions (and their bounding volumes) is about the same as the conventional SSI approach of subdivision methods. When two regular surfaces intersect tangentially, Ye and Maekawa [45] determine the topology of SSI curve using the Dupin indicatrices of the two surfaces. The osculating toroidal patches in the leaf nodes can serve for the same purpose as they capture the same second order local surface properties such as curvature. The simplicity of toroidal patches also makes the implementation of primitive geometric operations easy and robust.

The \( G^2 \)-continuous biarcs approximation to planar freeform curves was shown to be very useful not only in the acceleration of many geometric algorithms but also in the improvement of their robustness [23, 28, 29]. Liu et al. [32] employed the second order osculating torus approximation to freeform surfaces for the acceleration of point-projection algorithm. In recent work, Son et al. [44] accelerated the minimum distance computation between two surfaces of revolution using the osculating toroidal patches, which are generated by rotating the \( G^1 \)-biarc approximation to the profile curves of the rotational surfaces. In this paper, we consider the SSI problem using the osculating toroidal patches. The main advantage of using toroidal patches is in the simplicity of finding all binormal lines between two tori by computing the real roots of a polynomial equation of degree 8, which is also the main source of acceleration for the minimum distance algorithm of Son et al. [44].

3. BVH Construction

In this section, we show how to construct a BVH for a bicubic Bézier surface \( S(u,v), (0 \leq u, v \leq 1) \). The surface \( S(u,v) \) is first approximated by a rectangular array of bilinear surfaces \( L_j(u,v), (u,v) \in [u_{j-1}, u_j] \times [v_{j-1}, v_j] \), for \( i, j = 1, \cdots, 2^N \), where \( u_i = \frac{i}{2^N} \) and \( v_j = \frac{j}{2^N} \). Each bilinear surface \( L_j(u,v) \) is then approximated by a planar quadrange \( Q_j(u,v) \). The approximation error between \( Q_j(u,v) \) and the subpatch \( S_j(u,v) = S(u,v) \), on the subdomain \( [u_{j-1}, u_j] \times [v_{j-1}, v_j] \), is bounded by \( \epsilon_{ij} > 0 \).
Expanding the quadrangle $Q_{ij}$ by the distance $\epsilon_{ij}$, in all three dimensions, a bounding volume $V_{ij}$ is generated for the rectangular subpatch $S_{ij}$. Starting from the bounding volumes $V_{ij}$ at the leaf level nodes at the height $H$, we construct the BVH for the surface $S(u, v)$, in a bottom-up fashion.

### 3.1. Hierarchy of bilinear surfaces

For an efficient generation of the bicubic Bézier surface

$$S(u, v) = \sum_{\alpha=0}^{3} \sum_{\beta=0}^{3} b_{\alpha\beta} B_{\alpha}^3(u) B_{\beta}^3(v),$$

for $0 \leq u, v \leq 1$, where $b_{\alpha\beta}$ are the Bézier control points, we save the cubic Bézier basis function values precomputed at the uniform sample parameters $u_i = \frac{i}{3}$ and $v_j = \frac{j}{3}$ and reuse them repeatedly as needed in the rest of the SSI construction:

$$B_{\alpha}^3(u_i) = \frac{3!}{(3-\alpha)!\alpha!} (1-u_i)^\alpha u_i^{3-\alpha},$$

for $\alpha = 0, 1, 2, 3$, and $i = 0, \ldots, 2H$. These function values are reused for both $B_{\alpha}^3(u_i) = B_{\alpha}^3(u_{i-1})$ and $B_{\beta}^3(v_j) = B_{\beta}^3(v_{j-1})$.

The precomputation of Bézier basis functions accelerates the uniform sampling of the surface $S(u, v)$ at the parameters $(u_i, v_j)$, for $i, j = 0, \ldots, 2H$. Each subpatch $S_{ij}(u, v) = S(u, v)$, on the subdomain $[u_{i-1}, u_i] \times [v_{j-1}, v_j]$, can be approximated by a bilinear surface $L_{ij}(u, v)$ that interpolates the four corner points of $S_{ij}(u, v)$, which are among the uniform sampling points of $S(u, v)$:

$$L_{ij}(u, v) = \frac{u-u_{i-1}}{2H} \frac{v-v_{j-1}}{2H} S(u_{i-1}, v_{j-1})$$

$$+ \frac{u-u_{i}}{2H} \frac{v-v_{j-1}}{2H} S(u_i, v_{j-1})$$

$$+ \frac{u-u_{i-1}}{2H} \frac{v-v_{j}}{2H} S(u_{i-1}, v_j)$$

$$+ \frac{u-u_{i}}{2H} \frac{v-v_{j}}{2H} S(u_i, v_j). \quad (1)$$

The above uniform sampling points $S(u_i, v_j)$ also include the sampling points of $S(u, v)$ at lower resolutions. For example, when we replace the maximum height $H$ by a smaller value $h$, the corresponding uniform sample parameters $u_i = \frac{i}{3}$ and $v_j = \frac{j}{3}$ also belong to the original $(1+2H)$ sample parameters:

$$\hat{u}_i = \frac{i}{2h} = \frac{i \cdot 2H - h}{2H} = u_i 2^{h-3},$$

$$\hat{v}_j = \frac{j}{2h} = \frac{j \cdot 2H - h}{2H} = v_j 2^{h-3}.$$}

Consequently, there is no need of further resampling of the surface points $S(\hat{u}, \hat{v})$, for the approximation of $S(u, v)$ at lower resolutions.

The Bézier surface $S(u, v)$ is approximated by a hierarchy of bilinear surfaces, for a sequence of heights $h = 0, \ldots, H$. Filip et al. [9] showed that the bilinear approximation has a quadratic convergence, which means that the approximation error is reduced by 4-times each time we increase the height $h$ by one, to $h+1$.

In a similar way, we can approximate the surface normal $N(u, v) = S_u \times S_v$ by a hierarchy of bilinear normal surfaces, where $S_u$ and $S_v$ are the first partial derivatives of $S(u, v)$. The Gauss map of $S(u, v)$, on the subdomain $[u_{i-1}, u_i] \times [v_{j-1}, v_j]$, can be approximated by a spherical quadrangle with four corners $\hat{N}(u_{i-1}, v_{j-1}), \hat{N}(u_{i-1}, v_{j}), \hat{N}(u_i, v_{j-1}), \hat{N}(u_i, v_{j})$, on the unit sphere $S^2$, where $\hat{N}(u, v) = \hat{N}(u, v)/\|\hat{N}(u, v)\|$. Expanding the spherical quadrangle slightly by an amount estimated by Filip et al. [9], we can bound the Gauss map of $S(u, v)$ on the subdomain $[u_{i-1}, u_i] \times [v_{j-1}, v_j]$.

### 3.2. Hierarchy of planar quadrangles

The bilinear surfaces $L_{ij}(u, v)$ are quads in general. The interference test and the distance computation between two bilinear surfaces are more time-consuming than the case of two planar polygons in general position (i.e., two polygons not contained in the same plane). Moreover, the point-projection to bilinear surfaces (i.e., quads) is more difficult to compute than to planes. We approximate each quadratic surface patch $L_{ij}(u, v)$ by a planar quadrangle $Q_{ij}(u, v)$ and then by a rectangle $R_{ij}(u, v)$, so as to make the overlap test between two bounding volumes easier and thus faster. (The details of the overlap test will be discussed in Section [4]).

Given a bilinear surface $L_{ij}(u, v)$ of Equation (4) with four corners, the best plane approximation to $L_{ij}(u, v)$ is the tangent plane of $L_{ij}(u, v)$ at the midpoint:

$$L\left(\frac{u_{i-1} + u_i}{2}, \frac{v_{j-1} + v_j}{2}\right) = \frac{S(u_{i-1}, v_{j-1}) + S(u_{i-1}, v_j) + S(u_i, v_{j-1}) + S(u_i, v_j)}{4}.$$

Moving two corner points $S(u_{i-1}, v_{j-1})$ and $S(u_i, v_j)$ by

$$\mathbf{d} = -S(u_{i-1}, v_{j-1}) + S(u_i, v_{j-1}) + S(u_{i-1}, v_j) - S(u_i, v_j),$$

to

$$3S(u_{i-1}, v_{j-1}) + S(u_i, v_{j-1}) + S(u_{i-1}, v_j) - S(u_i, v_j),$$

and

$$-S(u_{i-1}, v_{j-1}) + S(u_i, v_{j-1}) + S(u_{i-1}, v_j) + 3S(u_i, v_j),$$

and similarly moving the other corner points $S(u_i, v_{j-1})$ and $S(u_{i-1}, v_j)$ by $-\mathbf{d}$, we construct the quadrangle $Q_{ij}(u, v)$ contained in the tangent plane at the midpoint of $L_{ij}(u, v)$.

Filip et al. [9] showed that the distance between the planar quadrangle $Q_{ij}(u, v)$ (which is also a parallelogram) and the subpatch $S_{ij}(u, v)$ can be bounded as follows:

$$\|S_{ij}(u, v) - Q_{ij}(u, v)\| \leq \frac{\sqrt{3} (\|S_u\| + 2\|S_v\| + \|S_v\|)}{8} = \delta_{ij},$$

where $\|v\| = \max(|v_1|, |v_2|, |v_3|)$ for a vector $v = (v_x, v_y, v_z)$, and $S_u, S_v, S_v, S_u, S_v$ are the second partial derivatives of a Bézier surface $S(u, v), (u, v) \in [0, 1] \times [0, 1]$, obtained by reparameterizing the subpatch $S_{ij}(u, v)$, $(u, v) \in [u_{i-1}, u_i] \times [v_{j-1}, v_j]$, as follows:

$$S_{ij}(\hat{u}, \hat{v}) = S\left(\frac{u_{i-1} + \hat{u}}{2H}, \frac{v_{j-1} + \hat{v}}{2H}\right),$$

for $0 \leq \hat{u}, \hat{v} \leq 1$. By expanding the planar quadrangle $Q_{ij}(u, v)$ by $\delta_{ij} > 0$, we can construct a bounding volume for the subpatch $S_{ij}(u, v)$. 

---

[4]: Reference to a section or subsection in the text.
[9]: Reference to a specific page or source in the text.
The osculating toroidal patch is taken sufficiently large enough to cover both the positions and the normal directions of the Bézier surface \( \hat{S}_{ij}(u, v) \), \((u, v) \in [0, 1] \times [0, 1] \). Depending on the surface curvature at the midpoint \( \hat{S}_{ij}(\frac{1}{2}, \frac{1}{2}) \), we take either the outer part or the inner part of the torus for approximating the surface of positive or negative curvature \([32]\).

The osculating toroidal patch is taken sufficiently large enough to cover both the positions and the normal directions of the Bézier surface \( \hat{S}_{ij}(u, v) \), \((u, v) \in [0, 1] \times [0, 1] \). Moreover, we take the boundary of the toroidal patch as the meridian and parallel circular arcs, which greatly simplifies geometric operations on the toroidal patch (Figure 3). For example, the Gauss map of such a toroidal patch is bounded by four circular arcs on the Gaussian sphere. Taking larger toroidal patches bounded by circular arcs, it is easy to test the non-overlap condition for their Gauss maps, which also implies a similar condition for the two Bézier surface patches approximated by the toroidal patches.

We can trim the osculating toroidal patches \( T_{ij} \) using non-circular boundary curves so that the Hausdorff distance between the trimmed patch and the Bézier surface \( \hat{S}_{ij}(u, v) \) is within a given error bound \( \epsilon > 0 \). This can be done by projecting the four corner points of \( \hat{S}_{ij}(u, v) \) to the toroidal patch and connecting the projected points to a quadrangle in the parameter space of the torus. By reparameterizing the quadrangle bilinearly, we can represent the trimmed toroidal patch as \( T_{ij}(u, v) \). Based on this parameter matching, we can bound the Hausdorff distance between \( \hat{S}_{ij} \) and \( T_{ij} \) by \( \max_{(u, v)} \| \hat{S}_{ij}(u, v) - T_{ij}(u, v) \| \). If the bound is larger than the given error bound \( \epsilon > 0 \), we can repeat the same procedure recursively by subdividing the Bézier surface \( \hat{S}_{ij}(u, v) \) into four pieces. Finally, we get a union of toroidal patches \( T_{ij} \), which may not even be connected to each other, but whose Hausdorff distance with the given Bézier surface \( S(u, v) \) is guaranteed to be within an error bound \( \epsilon > 0 \). By intersecting the toroidal patches, we approximate the SSI curve of the given freeform surfaces using the parameter matching between \( \hat{S}_{ij}(u, v) \) and \( T_{ij}(u, v) \).

### 4. SSI Algorithm

The BVH-based SSI algorithm proceeds by traversing the BVHs for the two freeform surfaces and detecting all pairs of leaf nodes whose bounding volumes overlap. For each pair of overlapping leaf nodes, we first intersect their osculating toroidal patches. When the two toroidal patches have no overlap in their Gauss maps, their SSI curve may have at most one branch with no bifurcation. Otherwise, the two surfaces may have tangential intersections, and we need further steps to determine the topological type of the SSI curve segments. In the final stage, we connect all the SSI curve segments thus constructed, in a correct topology.

#### 4.1. BVH traversal

Given two bicubic Bézier surfaces \( S_A(u, v) \) and \( S_B(s, t) \), let \( A \) and \( B \) denote their respective BVHs. The traversal of \( A \) and \( B \) is...
about the same as the sequence of subdivisions of $S_A$ and $S_B$ in the conventional SSI algorithms (that belong to the category of subdivision method). Instead of testing the overlap between the convex hulls of control points for the two surface patches to be intersected, we test the overlap between the bounding volumes stored in the interior nodes of $A$ and $B$ that correspond to the two surface patches.

When there is no overlap between the two bounding volumes, there is no intersection between the two surface patches under consideration. In the case of overlap, when $A$'s bounding volume is larger than $B$'s, we go down to the child nodes of $A$ and compare them with the current node of $B$; otherwise, we switch the roles of $A$ and $B$. At a leaf node of $A$ or $B$, we also switch the roles of $A$ and $B$. When we are at the leaf nodes of both $A$ and $B$, we move to the next stage of constructing SSI curve segments using the osculating toroidal patches stored in the leaf nodes of $A$ and $B$.

4.2. Construction of SSI curve segments

Given two leaf nodes of $A$ and $B$ and their osculating toroidal patches, it is easy to construct their Gauss maps and to test their overlap. When the Gauss maps have no overlap, the toroidal patches may have at most one branch in their SSI curve. In the case of having one branch, each endpoint of the SSI curve segment is located on the boundary of one toroidal patch $[4, 5, 6]$.

As discussed in Section 3.3, the osculating toroidal patches have their boundaries as circular arcs. By intersecting each of the boundary circular arcs against the other toroidal patch, we can find the two endpoints of the SSI curve branch. The SSI curve interior can then be approximated by a numerical tracing technique [2, 3, 4] or a specialized algorithm [15, 20, 33] for torus.

The intersection curve segments between two toroidal patches are then projected to the $uv$-parameter domain of the surface $S_A(u,v)$, using the parameter matching between $S_i(u,v)$ and $T_i(u,v)$ discussed at the end of Section 3.3 (The projection to the $st$-domain of the other surface $S_B(s,t)$ can be done in the same way.) There is one technical problem to consider here. Two adjacent trimmed toroidal patches $T_{i-1,j}(u,v)$ and $T_{i,j}(u,v)$ are not connected and they share no common boundary. (On the other hand, their counterparts $S_{i-1,j}(u,v)$ and $S_{i,j}(u,v)$ are smoothly connected on the same Bézier surface $S_A(u,v)$.) This means that the projections of the SSI curve segments from $T_{i-1,j}(u,v)$ and $T_{i,j}(u,v)$ to the $uv$-domain may not be connected across their common boundary line $u = u_i$. In the next subsection, we discuss how to deal with this problem by constructing a connected sequence of $G^1$-continuous biarcs in the $uv$-domain. The projected curve segments are approximated by $G^1$-biarcs within a given error bound $\varepsilon > 0$.

4.3. Merging SSI curve segments with $G^1$-biarcs

Each SSI curve segment in the $uv$-domain is usually given as a sequence of points $p_k, (k = 0, \ldots, N)$, representing a polyline. For the sake of simplicity, we consider the construction of a $G^1$-continuous $u$-monotone curve segment on a subdomain $[u_{i-1}, u_j] \times [v_{j-1}, v_j]$. We assume that the first point $p_0$ is located on the left boundary $u = u_{i-1}$ and the last point $p_N$ is on the line $u = u_i$. There may be a slightly different point $q_0$ (on the same line $u = u_i$) as a starting point for the next SSI curve segment on the subdomain $[u_i, u_{i+1}] \times [v_{j-1}, v_j]$ on the right. Thus we need to relocate $p_N$ to $p_N^*$, simply by averaging $p_N^* = (p_N + q_0)/2$, or for better, by numerically solving $S_i(u, v) = S_j(s, t)$. Then, we set $q_0^* = p_N^*$. The relocation $p_N^*$ is computed in a similar way.

When the SSI curves are generated, the tangent directions $v_k$ at the SSI points $p_k$ are also computed in the solution process. Using these $v_k$, we compute the tangent direction $v_k^*$ at the new location $p_N^*$, simply by averaging the directions for $p_N$ and $q_0$, or as a byproduct of the solution for $p_N^*$. In a similar way, we get a new direction $v_k^*$ at the new starting point $p_0^*$.

We construct a $G^1$-biarc that interpolates the two boundary conditions: $(p_0^*, v_0^*)$ and $(p_N^*, v_N^*)$ [15, 20, 33], and measure the maximum deviation of the intermediate points $p_k, (k = 1, \ldots, N-1)$. When the maximum deviation is larger than a given tolerance $\varepsilon > 0$, we repeat the approximation using two $G^1$-biarcs by interpolating an additional point $p_k$ and its tangent vector $v_k$. The point $p_k$ is taken, simply as the middle point, or for better, as the point of maximum deviation from the $G^1$-biarc approximation.

We relocate $p_m$ to a more accurate position $p_m^*$ and the corresponding tangent direction $v_m^*$. After that, we repeat the $G^1$-biarc constructions recursively in two subproblems.

In general, we need to consider the $G^1$-biarc construction for various different cases such as: (i) from a boundary point $p_0^*$ to an X-junction point $p_N^*$, or the other way around, (ii) from the leftmost point $p_0^*$ of a closed loop to the rightmost point $p_N^*$ of the same loop, and then back to $p_0^*$ around the other side of the loop, and so on. The one branch condition (on two transversally intersecting surface patches) greatly simplifies the ordering of the SSI points $p_k$, and thus the $G^1$-biarc approximation of these ordered points. Nevertheless, the ordering problem becomes more difficult when the surfaces intersect (almost) tangentially.

Even under the one branch condition, there is one serious technical problem in our approach – the osculating toroidal patches may not intersect even though their counterparts $S_i(u,v)$ have a branch of intersection curve. (This is because the one branch condition does not necessarily guarantee the existence of one branch.) For example, the sequence of SSI points $p_k, (k = 0, \ldots, N)$, may not be connected to another sequence of points $q_l, (l = 0, \ldots, N)$, in the subdomain on the right. Nevertheless, starting...
with the information \((p_1^*, v_1^*)\), we can do the conventional numerical curve tracing on the SSI curve: \(S_1(u, v) = S_2(s, t)\). But then, what if the whole loop is missing, by bad luck, without leaving any information \((p^*, v^*)\) to start with? Though the chance of missing a whole loop is extremely low, we may have (almost) tangential intersections of two surfaces when their Gauss maps overlap. We have not discussed this non-trivial case, yet.

The bottom line is that, regarding the issues of robustness and topological guarantees, we can do as much as other conventional methods do. The BVH-based approach can work with any other subdivision-based SSI methods, while incorporating their apparatus (such as ensuring at least one point on each branch or loop), and at the same time, improving the performance by using the pre-built subdivision structures for freeform surfaces. Thus, in the rest of this work, instead of trying to fill small gaps for the completeness of a typical SSI work, we focus on the main computational features that can be made possible using the precomputation of osculating toroidal patches and the nice geometric properties of \(G^1\)-biarc spline curves.

4.4. Measuring the SSI approximation error using \(G^1\)-biarcs

The \(G^1\)-biarc spline curve \((u(\theta), v(\theta))\) is only an approximate solution to the SSI curve, where the parameter \(\theta\) means an arc-length parameterization of the biarc spline curve. To test if the approximation error is within a given error bound \(\epsilon > 0\), we need to bound the maximum deviation of the curve-on-surface \(S_A(u(\theta), v(\theta))\) from the other surface \(S_B(s, t)\). For this purpose, using a similar \(G^1\)-biarc approximation of the SSI curve in the \(st\)-domain of \(S_B(s, t)\), we match pairs of circular arcs from both the \(uv\) and \(st\)-domains and estimate the upper bound for the squared distance function:

\[d^2(\theta) = |S_A(u(\theta), v(\theta)) - S_B(s(\theta), t(\theta))|^2,\]

by sampling on the angles \(\theta_i\) of the matching arcs and adding an error term \(\eta > 0\) estimated from Filip et al. to the maximum of the sampled function values, max \(d^2(\theta_i)\).

When the bounding condition: max \(d^2(\theta) + \eta < \epsilon^2\), is met, each of the matching SSI curves, \(S_A(u(\theta), v(\theta))\) and \(S_B(s(\theta), t(\theta))\), is guaranteed to be within a given tolerance \(\epsilon > 0\) from the other surface. The matching pair \((S_A(u(\theta), v(\theta)), S_B(s(\theta), t(\theta)))\) is considered to be an acceptable approximate solution to the SSI computation problem. Otherwise, we need to refine the \(G^1\)-biarc SSI curves in the \(uv\) and \(st\)-parameter domains by sampling some more points in the SSI curves if necessary.

5. Experimental Results

We have implemented the proposed SSI construction algorithm in C++, on an Intel Core i9-9900K 3.6GHz Windows PC with a 64GB main memory. To demonstrate the effectiveness of the BVH-based approach, we have tested the SSI algorithm on a large set of examples, including those of intersecting two almost identical surfaces.

The BVH construction on a bicubic Bézier surface typically takes less than one second in the preprocessing step, when the resolution of \(64 \times 64\) is used for sampling the \(uv\)-parameter domain of the surface. The construction time increases quadratically as we take finer resolutions along both \(u, v\)-directions. Thus the BVH construction would take less than one minute even for a high resolution of \(512 \times 512\). Nevertheless, there is a certain limitation on the maximum resolution, mainly due to the fixed memory space of a hardware system. On the other hand, the SSI computation takes far less time than the BVH construction.

Table 1: BVH storage for a bicubic Bézier surface (in MB).

<table>
<thead>
<tr>
<th>Height</th>
<th>Convex</th>
<th>AABB</th>
<th>RSS</th>
<th>Tori</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.599</td>
<td>0.112</td>
<td>0.142</td>
<td>0.157</td>
</tr>
<tr>
<td>5</td>
<td>6.374</td>
<td>0.447</td>
<td>0.493</td>
<td>0.605</td>
</tr>
<tr>
<td>6</td>
<td>25.414</td>
<td>1.792</td>
<td>2.217</td>
<td>2.334</td>
</tr>
<tr>
<td>8</td>
<td>413.189</td>
<td>28.724</td>
<td>36.548</td>
<td>37.773</td>
</tr>
</tbody>
</table>

Table 2: Construction time for complete BVH trees of \(H = 6\) (in seconds).

<table>
<thead>
<tr>
<th>Surfaces</th>
<th># Béziers</th>
<th>Convex</th>
<th>AABB</th>
<th>RSS</th>
<th>Tori</th>
</tr>
</thead>
<tbody>
<tr>
<td>GK 1</td>
<td>578</td>
<td>227.7</td>
<td>50.9</td>
<td>142.3</td>
<td>31.4</td>
</tr>
<tr>
<td>GK 2</td>
<td>578</td>
<td>212.7</td>
<td>50.6</td>
<td>140.6</td>
<td>35.0</td>
</tr>
<tr>
<td>GK 3</td>
<td>578</td>
<td>258.0</td>
<td>51.2</td>
<td>136.6</td>
<td>38.1</td>
</tr>
<tr>
<td>GK 4</td>
<td>2</td>
<td>0.6</td>
<td>0.2</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>GK 5</td>
<td>305</td>
<td>101.7</td>
<td>27.0</td>
<td>71.8</td>
<td>21.2</td>
</tr>
<tr>
<td>GK 6</td>
<td>273</td>
<td>106.8</td>
<td>23.9</td>
<td>64.5</td>
<td>18.0</td>
</tr>
<tr>
<td>Overlap</td>
<td>8</td>
<td>3.4</td>
<td>0.9</td>
<td>2.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Saddle 1</td>
<td>2</td>
<td>0.7</td>
<td>0.2</td>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td>Saddle 2</td>
<td>2</td>
<td>0.7</td>
<td>0.2</td>
<td>0.5</td>
<td>0.1</td>
</tr>
</tbody>
</table>
nodes when the BVH tree traversal is made all the way to the leaf level (i.e., without checking the one branch condition in the internal nodes). This way of complete traversal may produce a larger number of pairs than necessary, which is the reason why we check only with the height $H = 4$, until then not many early exits would be made at the higher levels $h = 1, 2, 3$. Though the AABB tree is often faster in the tree traversal (mostly for simple test examples) than the RSS tree, it is mainly due to the simple AABB overlap test, which takes less time than comparing two rectangles in general orientations in the $xyz$-space.

According to Table 3, the convex hull tree generates the tightest overlap tests, which saves much time wasted for unnecessarily checking the potential overlaps among surface subpatches. Though the RSS tree generates more redundant pairs than the convex hull tree, the overlap test for toroidal patches is simpler than for freeform surface patches. Considering all these, the RSS tree (combined with osculating toroidal patches) provides a good compromise for both space and time efficiency in the SSI computation for freeform surfaces.

In this paper, we have considered only bicubic Bézier surfaces. The same approach can be applied to Bézier surfaces of different degrees. The main difference is in the construction time, not in the BVH storage size. For higher degree, the number of control points increases, not only for the Bézier surface $S(u, v) = S_u \times S_v$, but also for the normal surface $N(u, v) = S_u \times S_v$. Table 4 compares Bézier surfaces of different degrees. Note that this result is independent of the shape of Bézier surfaces; i.e., the result is about the same for different Bézier surfaces of the same degree. On the other hand, the BVH traversal time for an SSI computation is irrelevant to the degrees; it is mainly dependent on the shapes and relative locations of the Bézier surfaces to be intersected.

Now, we consider a more serious issue, the robustness of our method, by generating some highly non-trivial test examples of freeform surfaces which intersect tangentially almost everywhere. Heo et al. [14] report some interesting results from such a test, where two almost coaxial cylinders were intersected, with the angle between the two cylinders being $\theta = 10^\circ, 1^\circ, 0.1^\circ, 0.01^\circ$. In Figure 7 of Heo et al. [14], the $X$-junctions are slightly missed at a small angle $\theta = 1^\circ$, and then completely missed at a smaller angle $\theta = 0.01^\circ$. We repeat the same test using our method. Figure 8 reports the test results, including some more challenging tests with $\theta = 0.0001^\circ, 0.00002^\circ$. The first row of Figure 7 shows a red cylinder approximated (within an error bound $\varepsilon = 10^{-10}$) by four bicubic polynomial Bézier surfaces, where the parallel lines are degree-elevated to cubic Bézier curves. Rotating the cylinder axis by angle $\theta$, we have generated blue cylinders for more and more difficult test examples. Two cylinders of the same radius intersect while sharing two exact tangential intersection points. In these test results, the tangential intersection curves are shown as yellow patches. The second row shows only the boundary curves of the yellow patches in the tangential intersection regions, and the zoomed views on their $X$-junctions are given in the third row. In Figure 8, a tiny shape change is made to the blue cylinders by rotating their bicubic Bézier surfaces about the axis of the cylinder by angle $45^\circ$. (Note that the maximum error in the Bézier surface approximation occurs at the almost tangential intersection points.) After that, we repeat the same tests, but the results are shown for a different set of angles $\theta$.

Table 5 reports the number of overlap pairs of leaf nodes in each of the cylinder examples shown in Figure 7. (The performance of AABB is highly dependent on the orientation of cylinder, which may explain the unusual result for the case of angle $\theta = 0.00002^\circ$. In this experiment, we mitigate the influence of orientation by taking the average of three tests using the axis directions along $(1, 0, 0), (1, 1, 0), (1, 1, 1)$. The ABB tree seems impractical in most of these test cases. On the other hand, the RSS tree looks impractical only after the angle gets smaller than $\theta = 0.0001^\circ$, where the overlap occurs almost everywhere in the two cylinders. Nevertheless, even in the case of $\theta = 0.00002^\circ$, using the osculating toroidal patches, we were able to detect the branching structure of the intersection curve reliably as shown in the rightmost example of Figure 7.

In theory, the convex hull tree might have generated a smaller number of overlap pairs; nevertheless, due to the large memory space required for the construction of intermediate convex hulls, we have experienced difficulty generating the convex hull tree for $H = 9$. As we have made some success in the challenging case of $\theta = 0.00002^\circ$, with more than 5M overlap pairs, any BVH tree combined with the osculating toroidal patches would also work for this test.

The yellow patches in Figure 7 are ideally the surface subpatches which are within a certain one-sided Hausdorff distance $\delta > 0$ to the other surface, the exact computation of which would require the intersection of a surface with the $\delta$-offsets of the other surface. This problem seems to be more difficult than the problem of intersecting two regular surfaces. In the current work, we take uniform samples from the osculating toroidal patch of one node to the osculating torus of the other overlap node, and

---

Table 3: Tree traversal time using BVH trees of $H = 6$ (in ms).

<table>
<thead>
<tr>
<th>Surfaces</th>
<th># Béz Pairs</th>
<th>Convex</th>
<th>AABB</th>
<th>RSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GK 1</td>
<td>(289,289)</td>
<td>134.07</td>
<td>39.14</td>
<td>44.35</td>
</tr>
<tr>
<td>GK 2</td>
<td>(289,289)</td>
<td>107.99</td>
<td>27.31</td>
<td>32.18</td>
</tr>
<tr>
<td>GK 3</td>
<td>(289,289)</td>
<td>102.45</td>
<td>26.31</td>
<td>29.25</td>
</tr>
<tr>
<td>GK 4</td>
<td>(1,1)</td>
<td>0.06</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>GK 5</td>
<td>(16,289)</td>
<td>6.85</td>
<td>1.89</td>
<td>1.77</td>
</tr>
<tr>
<td>GK 6</td>
<td>(81,192)</td>
<td>16.58</td>
<td>4.91</td>
<td>4.74</td>
</tr>
<tr>
<td>Overlap</td>
<td>(4,4)</td>
<td>2.64</td>
<td>0.48</td>
<td>0.80</td>
</tr>
<tr>
<td>Saddle 1</td>
<td>(1,1)</td>
<td>0.57</td>
<td>0.07</td>
<td>0.20</td>
</tr>
<tr>
<td>Saddle 2</td>
<td>(1,1)</td>
<td>0.89</td>
<td>0.08</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Table 4: Pairs of overlapping leaf nodes in complete BVH traversal with $H = 4$.

<table>
<thead>
<tr>
<th>Surfaces</th>
<th># Béz Pairs</th>
<th>Convex</th>
<th>AABB</th>
<th>RSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GK 1</td>
<td>(289,289)</td>
<td>12,008</td>
<td>71,533</td>
<td>16,367</td>
</tr>
<tr>
<td>GK 2</td>
<td>(289,289)</td>
<td>6,834</td>
<td>20,284</td>
<td>7,941</td>
</tr>
<tr>
<td>GK 3</td>
<td>(289,289)</td>
<td>3,547</td>
<td>6,276</td>
<td>5,034</td>
</tr>
<tr>
<td>GK 4</td>
<td>(1,1)</td>
<td>17</td>
<td>34</td>
<td>19</td>
</tr>
<tr>
<td>GK 5</td>
<td>(16,289)</td>
<td>1,228</td>
<td>4,009</td>
<td>1,748</td>
</tr>
<tr>
<td>GK 6</td>
<td>(81,192)</td>
<td>572</td>
<td>2,042</td>
<td>618</td>
</tr>
<tr>
<td>Overlap</td>
<td>(4,4)</td>
<td>522</td>
<td>1,049</td>
<td>572</td>
</tr>
<tr>
<td>Saddle 1</td>
<td>(1,1)</td>
<td>111</td>
<td>284</td>
<td>251</td>
</tr>
<tr>
<td>Saddle 2</td>
<td>(1,1)</td>
<td>125</td>
<td>286</td>
<td>250</td>
</tr>
</tbody>
</table>

Table 5: Construction time and BVH storage for different degrees of a Bézier surface (with $H = 6$).

<table>
<thead>
<tr>
<th>Surface</th>
<th>Time (ms)</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>RSS</td>
<td>Tori</td>
</tr>
<tr>
<td>2 x 2</td>
<td>183</td>
<td>51</td>
</tr>
<tr>
<td>3 x 3</td>
<td>244</td>
<td>68</td>
</tr>
<tr>
<td>4 x 4</td>
<td>314</td>
<td>79</td>
</tr>
</tbody>
</table>
collect those within a tolerance $\delta = 10^{-8}$ to the torus. The cylinder example is somewhat artificial as the cylinder is a simple surface by itself. In Figure\[C] we repeat the same robustness test to a saddle surface taken from the examples of Figure\[C].

6. Conclusions

In this paper, we have presented a new approach to the well-known SSI problem for freeform surfaces, which is based on a pre-built RSS tree with osculating toroidal patches stored in the leaf nodes. The high approximation order of osculating torus was shown to be an effective geometric tool for handling non-trivial cases of two freeform surfaces tangentially intersecting almost everywhere. In future work, we plan to develop a new technique that can bound the tangential intersection areas more precisely and estimate the approximation error more accurately.

Acknowledgments

We would like to thank our anonymous reviewers for their comments. This work was funded in part by the National Research Foundation of Korea (No. NRF-2019R1A2C1003490, NRF-2019K1A3A1A78112596), and in part by the Israel Science Foundation (grant No. 597/18).

References


Figure 5: Examples from Grandine and Klein [11]; the leftmost column shows the results of intersecting two freeform surfaces and the rightmost two columns show the intersection curves in the parameter domains of the red and blue surfaces, respectively.
Figure 6: (a) Two almost overlapping surfaces, and (b)–(c) two saddle surfaces.

Figure 7: Tangential intersection of two almost identical cylinders, where their two axes make a small angle $\theta = 0.01^\circ, 0.001^\circ, 0.0001^\circ,$ and $0.000002^\circ$, from left to right; the same tolerance $\delta = 10^{-8}$ was used for all test examples.
Figure 8: Rotating bicubic polynomial Bézier surfaces approximating (within a maximum error bound $\epsilon = 10^{-10}$) the red cylinder by angle $45^\circ$ about the axis of the cylinder and then intersecting with the blue cylinder, using the angles $\theta = 0.1^\circ, 0.01^\circ, 0.001^\circ, 0.0001^\circ$.

Figure 9: Tangential intersection of a saddle surface with its rotation about a normal line by a small angle $\theta = 0.01^\circ, 0.001^\circ, 0.0001^\circ, 0.00002^\circ$.