DOI: 10.1111/cgf.13950
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)
Volume 39 (2020), Number 2

Efficient Minimum Distance Computation for Solids of Revolution

S.-H. Son[1] S.-H. Yoon[2] M.-S. Kim[†1] G. Elber[3]

[1] Dept. of Computer Science and Eng., Seoul National University, South Korea
[2] Dept. of Multimedia Eng., Dongguk University, South Korea
[3] Computer Science Dept.,Technion, Israel

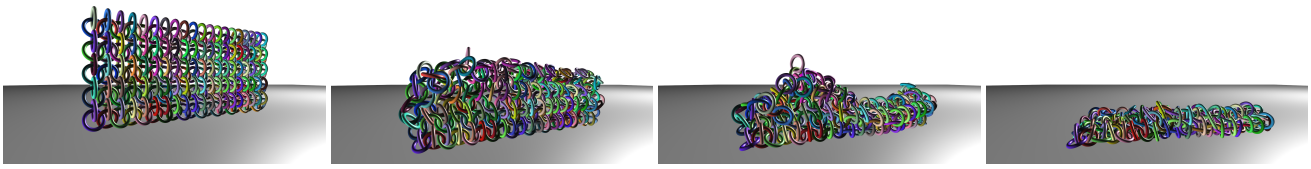

Figure 1: *Falling simulation of an interlocked ring mesh.*

Abstract

We present a highly efficient algorithm for computing the minimum distance between two solids of revolution, each of which is defined by a planar cross-section region and a rotation axis. The boundary profile curve for the cross-section is first approximated by a bounding volume hierarchy (BVH) of fat arcs. By rotating the fat arcs around the axis, we generate the BVH of fat tori that bounds the surface of revolution. The minimum distance between two solids of revolution is then computed very efficiently using the distance between fat tori, which can be boiled down to the minimum distance computation for circles in the three-dimensional space. Our circle-based approach to the solids of revolution has distinctive features of geometric simplification. The main advantage is in the effectiveness of our approach in handling the complex cases where the minimum distance is obtained in non-convex regions of the solids under consideration. Though we are dealing with a geometric problem for solids, the algorithm actually works in a computational style similar to that of handling planar curves. Compared with conventional BVH-based methods, our algorithm demonstrates outperformance in computing speed, often 10–100 times faster. Moreover, the minimum distance can be computed very efficiently for the solids of revolution under deformation, where the dynamic reconstruction of fat arcs dominates the overall computation time and takes a few milliseconds.

CCS Concepts

• ***Computing methodologies*** → *Minimum distance computation; Solid of revolution; Bounding volume hierarchy; Fat arc; Toroidal patch; Deformation; Acceleration;*

1. Introduction

Efficient minimum distance computation is an important first-step for the acceleration of many geometric operations on solid models. Conventional algorithms are efficient in handling convex objects [Gilbert et al. (1988), Lin and Canny (1991)]. General non-convex objects are often represented in a hierarchy of convex bounding volumes [Quinlan (1994)]. The convex volumes are easy to handle; however, they are ineffective in approximating non-convex parts of interacting objects in close proximity, such as the interlocked rings in Figure 1. This limitation motivates the introduction of non-convex bounding volumes to the BVH structure.

In the planar case, non-convex bounding regions such as fat arcs (FA) and bounding circular arcs (BCA) are highly effective in accelerating geometric algorithms (such as intersection [Sederberg et al. (1989)], offset [Lee et al. (2015a)], medial axis, and Voronoi diagram computations [Lee et al. (2016)]) for planar freeform curves. (Fat arc is an expansion of arc.) The efficiency of these regions (bounded by circular arcs) is based on their higher approximation order to planar curves. Moreover, the circular arcs are relatively easy to handle. As a natural extension of fat arcs to the three-dimensional case, [Krishnan et al. 1998a] suggested a spherical

† Corresponding author: mskim@snu.ac.kr

shell as a bounding volume. The approximation order of spherical patches to non-convex surfaces is low. On the other hand, toroidal patches have a higher approximation order to arbitrary surfaces [Liu et al. (2009)]. Thus we consider an extension of fat arcs to fat toroidal patches.

In this paper, using the BVH of fat toroidal patches, we develop an efficient algorithm for computing the minimum distance between two moving solids of revolution. The surface of revolution is extensively used for the design of many industrial parts and graphical models, including the cutter tools for 5-axis CNC flank machining [Bo and Barton (2019)], the lid and body parts of the Utah Teapot, and the volume interiors filled with trusses and microstructures for fabrication (3D printing).

Figure 2 shows a solid of revolution generated by rotating a profile curve $C(t) = (0, y(t), z(t))$ in the yz-plane around the z-axis. The planar curve can be approximated by a connected sequence of G^1-continuous circular arcs within an arbitrary error bound $\varepsilon > 0$ [Sir et al. (2006)]. By rotating these arcs, the boundary surface of revolution can be approximated by G^1-continuous toroidal patches within the same bound $\varepsilon > 0$ (Figure 3). The distance between two solids bounded by these toroidal patches provides a good approximate solution to the minimum distance within an error bound of $2\varepsilon > 0$. The whole problem is thus essentially reduced to that of computing the distance between toroidal patches. To make distance computation easy, we subdivide toroidal patches to elliptic or hyperbolic pieces, by segmenting the planar profile curve into z-monotone convex and concave pieces. (Related technical details are in Section 4.1.) The arcs for the convex/concave side of a profile curve generate toroidal patches on the convex/concave side of their respective tori.

[Sir et al. (2006)] explains how to construct a *biarc* (consisting of two G^1-continuous circular arcs) that interpolates the endpoints and the end tangent directions of a curve segment. When the approximation error is larger than $\varepsilon > 0$, we subdivide the curve segment (at the mid parameter) into two pieces and repeat the same construction for each piece recursively. The hierarchy of recursive subdivisions of the profile curve generates a balanced binary tree, where each node represents a *fat biarc* (which is the expansion of a biarc by an approximation error to the corresponding curve segment). The fat-biarc tree also represents a bounding volume hierarchy (BVH) for the surface of revolution, where each node can be interpreted as the rotated volume of a fat biarc.

In Section 3, we show that the distance between two toroidal patches can be computed using a simple formula: $l \pm r_1 \pm r_2$, where l is the length of a certain *binormal* line segment orthogonal to both the major circles of the two tori, r_i's are the minor circles' radii, and the $\pm$ signs are determined based on the convexity and the inward/outward surface orientation of the toroidal patches. This is based on the fact that each binormal line to two toroidal patches T_1 and T_2, also pass through their major circles orthogonally. Our circle-based method is more efficient than an alternative approach of computing the binormal lines of two toroidal patches $T_1(u,v)$ and $T_2(s,t)$, interpreted as bivariate surfaces.

The biarc tree construction takes more time than the minimum distance computation itself. Nevertheless, once built in a preprocessing step, the static structure can be used repeatedly for the

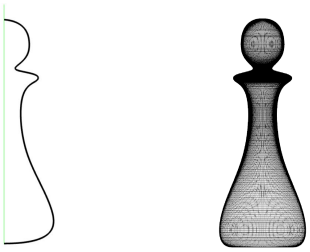

Figure 2: *A profile curve for a solid of revolution.*

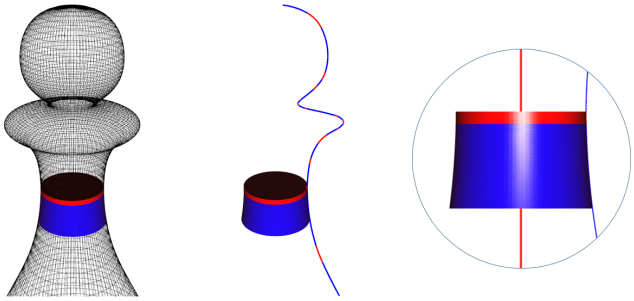

Figure 3: *A surface of revolution approximated with G^1-continuous toroidal patches (shown in the left) generated by rotating a G^1-biarc (in the middle) around the axis, and a zoomed view of the two smoothly connected patches (in the right).*

solids at all different poses. For the case of deformable solids of revolution, however, the profile curves change shape dynamically and we need to reconstruct their biarc trees on the fly. However, the reconstruction of the biarc tree is considerably more efficient than other conventional BVH structures for general bivariate surfaces or polygonal mesh models. For the test solid models of deformation used in the experiments of this paper, we have constructed the biarc tree and the BVH of toroidal patches in only a few milliseconds, while supporting real-time performance.

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

- We propose an algorithm for computing the minimum distance between two solids of revolution, which outperforms other conventional algorithms in computing speed.
- The improvement is based on the high approximation order of toroidal patches to the surfaces of revolution and the geometric simplicity of toroidal patches in measuring distances.
- The BVH of toroidal patches can be represented in a highly compact way using a biarc tree, which is a hierarchical data structure for the profile curve, not for the surface of revolution.
- By dynamically reconstructing the biarc tree for profile curves, we can support a highly efficient algorithm for computing the distance between two deforming solids of revolution.

2. Related Work

There are many efficient algorithms for computing the minimum distance between two convex objects. In particular, [Gilbert et al. (1988), Lin and Canny (1991)] are known to be two of the most efficient algorithms for this problem, including other variants for further improving the computing speed of these algorithms [Cameron (1997), van den Bergen (1999)]. [Lin and Manocha (2004)] reports an extensive survey on collision detection and distance computation.

[Larsen et al. 1999] proposed a general paradigm for computing the minimum distance between non-convex objects represented in the BVH of sphere-swept bounding volumes. They maintain a priority queue of pairs of BVH nodes, where each BVH node is the root for a subtree and represents a volume that bounds a subset of the object. [Johnson and Cohen (1998), Johnson (2005)] took a similar approach to compute the minimum distance between two B-spline surfaces. The main difference is in a dynamic construction of bounding volumes based on the minimum distance locations between the convex hulls of the B-spline control points for the surface patches under consideration. [Chang (2011)] presented a different way of splitting the Bézier surfaces under consideration. [Kim et al. (2011)] introduced a preprocessing technique that can speed up a dynamic construction of simple bounding volumes for B-spline surfaces. These conventional methods employed only convex bounding volumes.

In the planar case, [Sederberg et al. (1989)] accelerated the curve-curve intersection for planar Bézier curves by using a BVH of *fat arcs* which are non-convex regions generated by offsetting circular arcs by the approximation errors to the given planar curve. Based on the fat-arc BVH, [Kim et al. (2012)] demonstrated highly improved performance in the efficiency as well as the robustness in trimming the planar curve offset self-intersections. [Lee et al. (2015a)] further improved the efficiency of planar offset trimming by using a BVH of bounding circular arcs (BCA). [Meek and Walton (1995)] introduced the concept of BCA for the purpose of proving the cubic convergence of their biarc approximation to planar curves. Based on the BCA-tree, [Lee et al. (2016)] developed a highly robust real-time algorithm for computing the medial axis and Voronoi diagram for planar curves.

Fat arcs are often used in the form of fat biarcs as can be seen in many applications [Lin and Rokne (2002), Yong et al. (2006)]. A biarc (with two G^1-continuous circular arcs) can interpolate both the positions and the tangent directions at two endpoints of a curve segment at the same time. Thus it is easier to handle one biarc rather than two arcs separately. In this paper, when we use the term arc tree, we mean a biarc tree representation, but each arc in a node is processed individually and generates a separate toroidal patch.

The spherical shell of [Krishnan et al. 1998a] is the first non-convex bounding volume developed for surface models such as polygonal meshes. Each spherical shell is constructed with two concentric spheres and a cone with the apex at the sphere center. Because of the spherical shape, the spherical shell has limitations in bounding hyperbolic surfaces such as saddle surfaces. In the point projection problem to a freeform surface, which is a special type of minimum distance computation between a point and a surface, [Liu et al. (2009)] employed a sequence of osculating tori to a surface for the acceleration of point projection. An osculating circle/torus provides the best approximation around a specific point. Nevertheless, a G^1-biarc provides a better approximation over a curve segment because of the interpolation power at both endpoints of the curve, including the positions and the tangent directions. This property also justifies the use of toroidal patches for bounding the surfaces of revolution in the current work.

[Neff (1990)] introduced an analysis of the algebraic complexity of the circle-circle distance problem in space and showed that the minimum distance is given as a root of polynomial of degree 8. Nevertheless, the minimum distance itself can be computed more efficiently than by the approach of numerically solving a polynomial equation of degree 8. [Vranek (2002)] reduced the problem to an efficient numerical search for the global minimum of a distance function, where the search is guaranteed to find the global minimum by solving an auxiliary polynomial equation of degree 4. The analysis of [Neff (1990)] is still quite important – it is related to a more general problem of finding all binormal lines to two circles. In Section 3, we present an elementary construction for all binormal lines (and their intersection points) to two circles in space. [Eberly (2007)] also derived the same polynomial equation, but purely algebraically. Our derivation is geometric, which is more useful in the acceleration of search for specific binormal lines needed by our algorithm.

3. Distance Computation for Toroidal Patches

3.1. Torus Parameterization

A torus is generated by rotating a circle of radius r about an axis line in the plane of the circle and at a distance $R > r$ from the circle center. For the sake of simplicity, we may assume that the z-axis is the rotation axis and the circle of radius r is contained in the yz-plane as follows:

$$(0, R + r\cos\phi, r\sin\phi),$$

for $0 \le \phi \le 2\pi$. The torus $T(\phi,\theta)$ can be represented in parametric form as

$$((R + r\cos\phi)\cos\theta, (R + r\cos\phi)\sin\theta, r\sin\phi),$$

for $0 \le \phi, \theta \le 2\pi$. The outward unit surface normal $N(\phi,\theta)$ is given in a simple form

$$(\cos\phi\cos\theta, \cos\phi\sin\theta, \sin\phi).$$

By restricting $\underline{\phi} \le \phi \le \overline{\phi}$, a toroidal patch is generated by rotating a circular arc around the z-axis.

The torus T in an arbitrary position and orientation can be obtained by rotating and translating the standard torus T as follows:

$$\mathbf{u}\,(R + r\cos\phi)\cos\theta + \mathbf{v}\,(R + r\cos\phi)\sin\theta + \mathbf{n}\,r\sin\phi + \mathbf{t},$$

where $[\mathbf{u}\ \mathbf{v}\ \mathbf{n}]$ is a rotation matrix and $\mathbf{t}$ is a translation vector.

3.2. Binormal lines to the major circles

When two toroidal patches share a common binormal line, this line is also a binormal line to the major circles of the two tori. The first step to compute the minimum distance between two toroidal

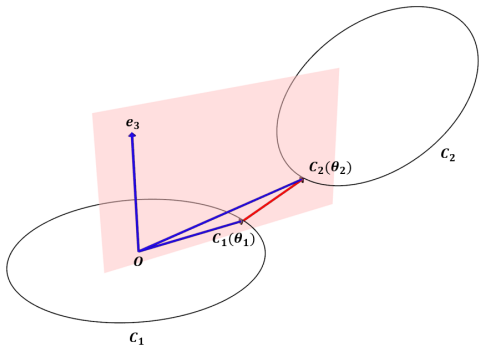

Figure 4: *Coplanarity condition for three vectors (in blue).*

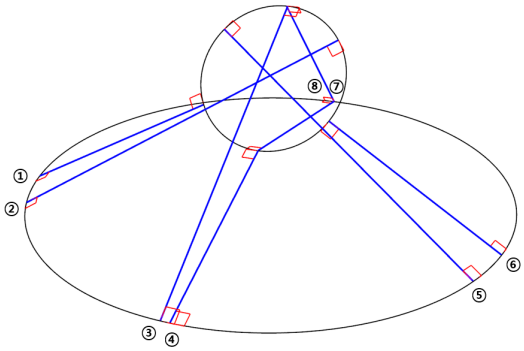

Figure 5: *Eight binormal lines to two circles in space.*

patches is thus to compute the binormal lines to the major circles, the maximum number of which is known to be 8.

Given two toroidal patches T_1 and T_2, we assume that the first torus $T_1(\phi_1,\theta_1)$ is given in a standard form:

$$((R_1+r_1\cos\phi_1)\cos\theta_1,(R_1+r_1\cos\phi_1)\sin\theta_1,r_1\sin\phi_1),$$

for $\underline{\phi_1} \le \phi_1 \le \overline{\phi_1}$, and $0 \le \theta_1 \le 2\pi$, and the second torus $T_2(\phi_2,\theta_2)$ is given in a general form:

$$\mathbf{u}\,(R_2+r_2\cos\phi_2)\cos\theta_2+\mathbf{v}\,(R_2+r_2\cos\phi_2)\sin\theta_2+\mathbf{n}\,r_2\sin\phi_2+\mathbf{t},$$

for $\underline{\phi_2} \le \phi_2 \le \overline{\phi_2}$, and $0 \le \theta_2 \le 2\pi$.

A binormal line connecting two points on the tori $T_1(\phi_1,\theta_1)$ and $T_2(\phi_2,\theta_2)$ also passes through their major circles C_1 and C_2 at the same parameter locations of θ_1 and θ_2:

$$C_1(\theta_1) = (R_1\cos\theta_1, R_1\sin\theta_1, 0), \text{ and}$$
$$C_2(\theta_2) = \mathbf{u}\,R_2\cos\theta_2+\mathbf{v}\,R_2\sin\theta_2+\mathbf{t}.$$

Now the difference vector $C_2(\theta_2)-C_1(\theta_1)$ (parallel to the binormal line) is orthogonal to the circle C_1, and consequently this vector is parallel to the normal plane of C_1 at $C_1(\theta_1)$ (which contains the z-axis and the origin $\mathbf{0}$ as shown in Figure 4). This means that three vectors $C_1(\theta_1)$, $C_2(\theta_2)-C_1(\theta_1)$, and $\mathbf{e}_3$ are coplanar, which implies that three vectors $C_1(\theta_1)$, $C_2(\theta_2)$, and $\mathbf{e}_3$ are also coplanar. (Note that $C_1(\theta_1)$ and $C_2(\theta_2)$ denote the difference vectors: $C_i(\theta_i)-\mathbf{0}$, $i=1,2$.) In Appendix A, this coplanarity condition is converted to a multivariate polynomial equation in four variables $\alpha_i=\cos\theta_i$ and $\beta_i=\sin\theta_i$, $i=1,2$. Switching the roles of T_1 and T_2 and interpreting the coplanarity condition in the local coordinates of T_2, we can derive a similar equation in α_i,β_i. Together with two additional constraints: $\alpha_i^2+\beta_i^2=1$, the coplanarity conditions can be reduced to one univariate polynomial equation of degree 8. Appendix A presents an elementary construction for all these details. Figure 5 shows an example of two circles with 8 binormal lines.

3.3. Directions for binormal lines

We consider how to accelerate the search for binormal lines by restricting their directions. Figure 6 shows two toroidal patches T_1 (in red) and T_2 (in blue) and the Gauss maps of T_1 and $-T_2$ on the unit sphere S^2 (shown as a wireframe). Note that we take the reversed

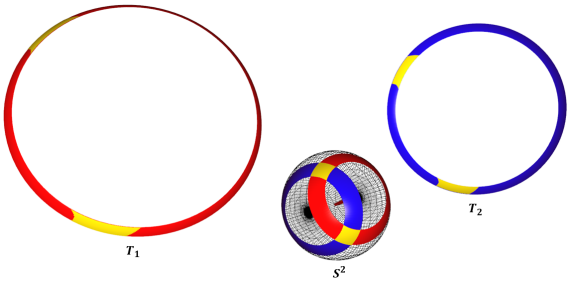

Figure 6: *Areas of toroidal patches that may admit binormal lines.*

normal directions of T_2 by computing the Gauss map of $-T_2$. This is because the outward normals of T_1 and T_2 are the opposite at their nearest points $\mathbf{p}\in T_1$ and $\mathbf{q}\in T_2$.

When the two Gauss maps of T_1 and $-T_2$ have no overlap on S^2, the two patches T_1 and T_2 admit no binormal line in their surface interiors. Otherwise, they have a non-empty overlap $G\subset S^2$ (in yellow on the unit sphere). Let $\hat{T}_1$ and $-\hat{T}_2$ denote the subpatches (in yellow on T_1 and T_2), as the result of inverse mappings of G to T_1 and $-T_2$. The nearest points $\mathbf{p}\in T_1$ and $\mathbf{q}\in T_2$ can only be located on the subpatches $\hat{T}_1$ and $\hat{T}_2$.

The subpatches $\hat{T}_1$ and $\hat{T}_2$ can be projected to arcs $\hat{C}_1$ and $\hat{C}_2$ on their respective major circles C_1 and C_2; they essentially bound the ranges of θ_i, $\alpha_i=\cos\theta_i$, $\beta_i=\sin\theta_i$, $(i=1,2)$. In the restricted ranges, we solve the polynomial equation of degree 8 very efficiently, using a recent technique of [Machchhar and Elber (2016)] that can compute the real roots of univariate polynomials about 10 times faster than other conventional methods.

4. Biarc Trees and Priority Queue

4.1. Biarc trees

We assume that each solid of revolution is generated by rotating a planar profile curve $C(t)=(0,y(t),z(t))$ (in the positive yz-halfplane with $y\ge 0$) around the z-axis. The profile curve is further segmented into monotone convex/concave pieces by subdividing at each of y- and z-extreme points (by solving $y'(t)=0$ and $z'(t)=0$) and inflection points (by solving $y'(t)z''(t)-y''(t)z'(t)=0$). Each

monotone piece is then further segmented into curvature monotone smaller pieces (called *spiral* curves) by solving $\kappa'(t) = 0$, where $\kappa(t)$ is the curvature function of $C(t)$.

The planar curve $C(t)$ is then given as a connected sequence of spirals, each of which can be bounded by an AABB (axis-aligned bounding box). In the higher levels, the curve $C(t)$ is represented in an AABB tree, and the solid of revolution is then bounded by a hierarchy of truncated cylinders.

Each spiral curve segment is stored in the leaf level of the AABB tree. From each leaf node of the AABB tree, we build a biarc tree by recursively subdividing the spiral curve at the mid parameter until the approximation error is reduced within a given error bound $\varepsilon > 0$ (see [Sir et al. (2006)]). In each node of the biarc tree, we also store the line segment that best approximates the biarc for the node. This line segment generates a truncated cone, which can accelerate the minimum distance computation for internal nodes. (See Algorithms 1 and 2.)

The biarc approximation error can be estimated very efficiently using the thickness of bounding volumes for spirals. [Kumosenko (2013)] developed a tight bounding volume for spiral, *bilens*, whose boundary is composed of two biarcs, each interpolating both the two endpoints and the two end tangent directions of a spiral curve segment. In this paper, we employ the bilens for an efficient and tight bounding of the biarc approximation error.

Algorithm 1: BVH building algorithm for a Spiral

```
Result: BVH for a spiral
input: Precision ε > 0
Approximate with a biarc;
Approximate with a line segment;
Build toroidal patches by rotating biarc;
Build conical patch by rotating line segment;
if Biarc approximation error < ε then
    Set leaf flag;
else
    Subdivide spiral into two pieces at the mid parameter;
    Call this routine recursively for each subdivided piece;
    Set child nodes;
end
```

4.2. Priority queue and lower/upper bounds

The conventional BVH-based algorithms for minimum distance computation follow the same algorithmic flow as originally introduced by [Larsen et al. 1999], where certain valid pairs of BVH nodes are maintained in a priority queue. The priority of each pair is set to the minimum distance between the bounding volumes in the BVH nodes of the pair. The priority queue is initialized with a single element – the pair of BVH root nodes. The algorithm maintains a lower bound $\underline{d}$ and an upper bound $\overline{d}$ for the minimum distance d, and the priority queue keeps only the pairs with their distances being values within the two bounds. Iteratively updating the priority queue and the two bounds, the algorithm reduces the difference $\overline{d} - \underline{d}$ within a given tolerance. The minimum distance is then set to the average $d = (\overline{d} + \underline{d})/2$.

We also follow the same procedure. Nevertheless, due to the special geometric structure of our toroidal bounding volumes, there are slight differences in computing the two bounds: $\overline{d}$ and $\underline{d}$. [Larsen et al. 1999] takes the upper bound as $\overline{d} = \min\{\|\mathbf{p}_i - \mathbf{q}_j\|\}$, where $\mathbf{p}_i$'s and $\mathbf{q}_j$'s are some sampling points from each of the two solids. On the other hand, we take an upper bound $\overline{d} = \min\{\|\hat{\mathbf{p}}_i - \hat{\mathbf{q}}_j\| + \varepsilon_i + \varepsilon_j\}$, where $\hat{\mathbf{p}}_i$ and $\hat{\mathbf{q}}_j$ are the nearest points on two toroidal patches under consideration for the pair of BVH nodes and ε_i and ε_j are their approximation error bounds. Because of the special structure of our biarc approximation, which guarantees two-sided Hausdorff distance bounding, there are a pair of points $\mathbf{p}_i$ and $\mathbf{q}_j$ on the original solids, where $\|\mathbf{p}_i - \hat{\mathbf{p}}_i\| < \varepsilon_i$ and $\|\hat{\mathbf{q}}_j - \mathbf{q}_j\| < \varepsilon_j$. This implies that $\|\mathbf{p}_i - \mathbf{q}_j\| < \|\hat{\mathbf{p}}_i - \hat{\mathbf{q}}_j\| + \varepsilon_i + \varepsilon_j$. There is a slight overestimation in our upper bound $\overline{d}$. But this approach makes the algorithm efficient by skipping the sampling step on the original boundary surfaces. Moreover, we compute only the distance $\|\hat{\mathbf{p}}_i - \hat{\mathbf{q}}_j\|$, not the points $\hat{\mathbf{p}}_i, \hat{\mathbf{q}}_j, \mathbf{p}_i, \mathbf{q}_j$. (See Algorithm 2.)

Algorithm 2: BVH algorithm for distance computation

```
Result: Minimum distance
input  : BVH A, B
output: Upper Bound
Upper Bound = ∞;
Insert (Root A, Root B) into priority queue Q;
while Q is not empty do
    Pop (Node A, Node B) from Q;
    if Node A == leaf & Node B == leaf then
        d = Distance between toroidal patches;
        ub = d + ε_A + ε_B;
        if ub < Upper Bound then
            Upper Bound = ub;
        end
    else
        d = Distance between conical/cylinder patches;
        lb = d − ε_A − ε_B;
        if lb < Upper Bound then
            Insert pairs of children into Q with lb;
        end
    end
end
```

In a similar way, we take the lower bound: $\underline{d} = \min\{\|\hat{\mathbf{p}}_i - \hat{\mathbf{q}}_j\| - \varepsilon_i - \varepsilon_j\}$, where the minimum is taken for all pairs of BVH nodes in the priority queue. In fact, the minimum is stored in the top element of the priority queue, i.e., the pair of BVH nodes with the highest priority. Note that the priority is set to the minimum distance between the bounding volumes for the BVH nodes, which is the same as $\|\hat{\mathbf{p}}_i - \hat{\mathbf{q}}_j\| - \varepsilon_i - \varepsilon_j$. The rest of the algorithm is essentially the same as that of [Larsen et al. 1999].

The distance computation for toroidal patches is more expensive than for cylindrical or conical patches. When dealing with BVH internal nodes, we may replace the toroidal patches by cylindrical or conical patches to speed up the algorithm. The main disadvantage of these simple patches is in their lower approximation order to the original surface, which results in a higher chance of unnecessarily comparing many redundant pairs of nodes. Nevertheless,

for the pairs of BVH nodes around the top levels, comparison with these simple patches often very efficiently discards the majority of redundant pairs of BVH nodes.

5. Experimental Results

We have implemented our algorithm in C++ on an Intel Core i7-6700 3.4GHz PC with a 16 GB main memory. To demonstrate the effectiveness of our approach, we have tested our algorithm on five static solids of revolution (Figure 7) and one deformable solid of revolution (Figure 10). For each pair of solid models (M_i, M_j), $(i \leq j)$, we computed the minimum distance $d(M_i, M_j)$. By interpolating the profile curves for the five static solids of revolution, starting from the pawn model in the upper left corner of Figure 7 and going around in the counter-clockwise order, we have also generated a deformable solid model of revolution, denoted as M_t, some snapshots of which are shown in Figure 10.

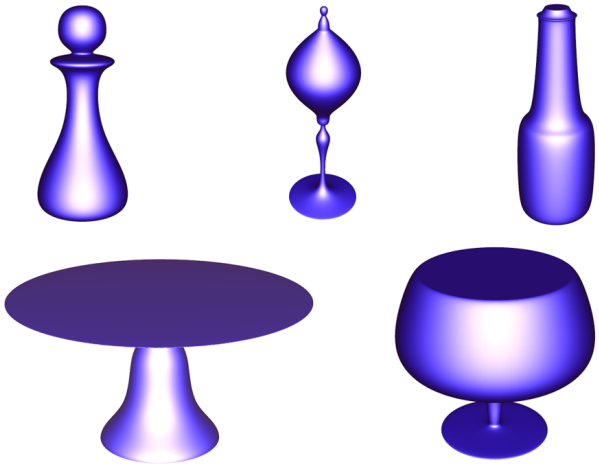

Figure 7: *Five static solids of revolution used for the experiments.*

For each model M_i, we have computed the minimum distance $d(M_i, M_{t_j})$, for every sample of M_{t_j}, taken from a total of 1000 discrete frames of M_t. Figure 8 shows the average computation time for the 1000 sample pairs (M_i, M_{t_j}), $(j = 1, \cdots, 1000)$. Compared with the minimum distance computation, the BVH construction is a far more time-consuming process, which is usually done in a preprocessing step. For the majority of real-time applications for deformable models, dynamic reconstruction of their BVH structures is the main bottleneck in the overall computation. Based on an efficient G^1-biarc approximation for the profile curves, our approach outperforms other methods in this respect. Figure 9 shows the construction time for the BVH of each solid model, where the last one is the average time for building each of the 1000 BVHs for M_{t_j}. This result means that, even with a high precision $\varepsilon = 10^{-5}$, we can support real-time interference tests for many applications handling up to several deformable solid models of revolution, which is known to be an extremely difficult computational task in conventional methods.

For static models, we can make some quantitative comparisons of our approach against conventional methods, represented

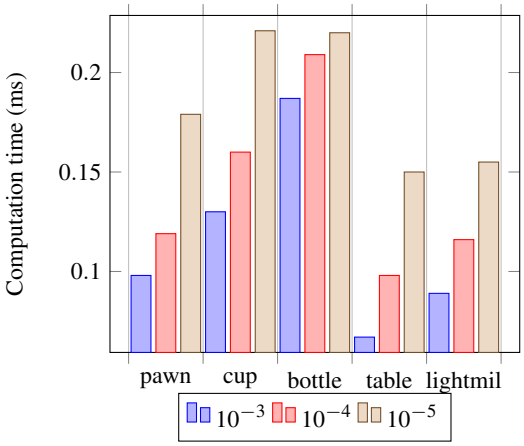

Figure 8: *Average computation time (in ms) for the minimum distance with a deformable model M_t.*

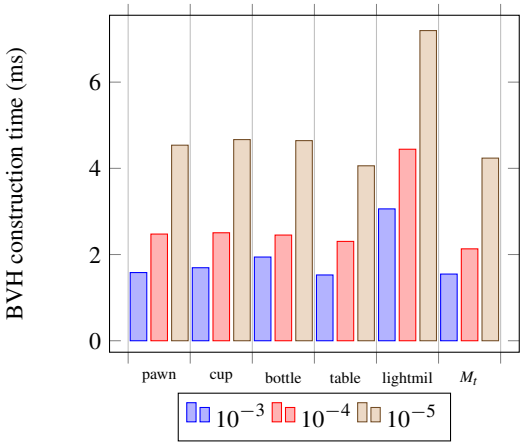

Figure 9: *BVH construction time (in ms).*

by the algorithm of [Larsen et al. 1999] using the PQP library (http://gamma.cs.unc.edu/SSV/). In Tables 1–2, the columns 'PQP' give the results from the experiments using the PQP library, and the columns 'Ours' report the test results from the algorithm proposed in this paper.

Table 1 compares the storage sizes of the two different approaches, where the biarc-based representation takes considerably less memory (often more than 1000 times) than the conventional methods. Table 2 shows a similar comparison of the numbers of BVH nodes. The biarc-approximation has a cubic convergence rate, which means that each time we subdivide a curve into two, the error will be reduced $8(= 2^3)$ times. On the other hand, the sphere-swept bounding volumes of [Larsen et al. 1999] and the oriented bounding box of [Gottschalk et al. (1996)] have only quadratic convergence. The higher-order convergence to a univariate curve (i.e., the profile curve), not directly to a bivariate surface, is the main source for the outperformance of our approach in the data reduction. (See [Lee et al. (2015b)] for comparisons among bounding

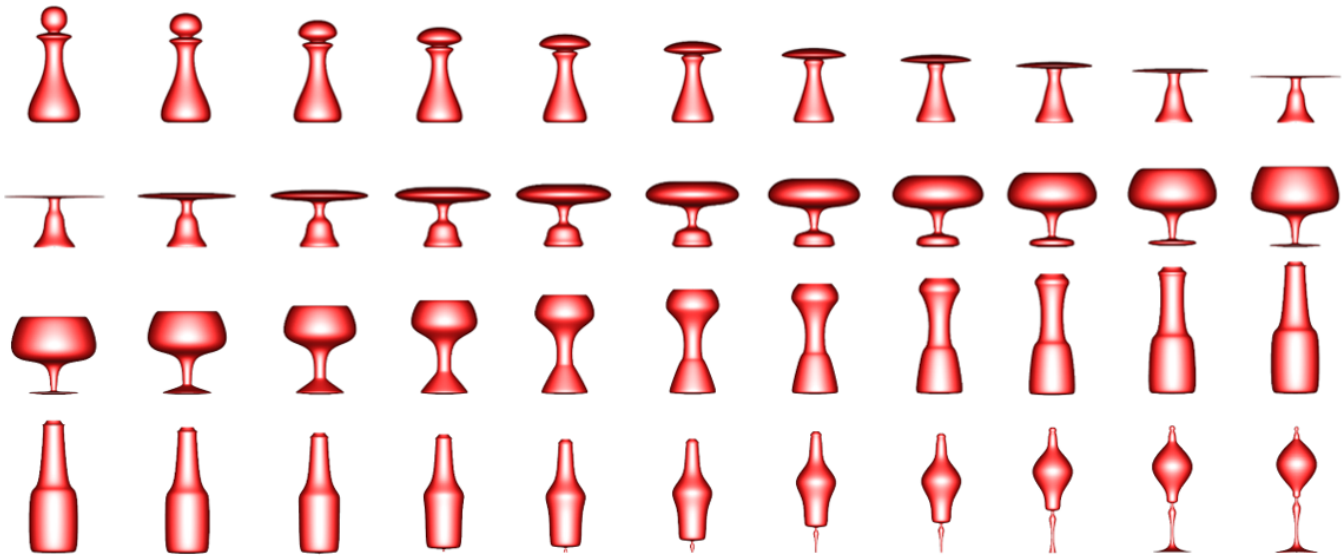

Figure 10: *Snapshots from a deformable solid of revolution.*

Table 1: *Comparison of BVH data sizes (in MB).*

Precision		10^{-3}		10^{-4}		10^{-5}
Models	PQP	Ours	PQP	Ours	PQP	Ours
pawn	62	0.081	550	0.203	1437	0.527
cup	59	0.096	657	0.199	1294	0.474
bottle	66	0.099	593	0.191	2003	0.506
table	62	0.079	674	0.181	1395	0.415
lightmil	91	0.187	982	0.347	590	0.776

Table 2: *Comparison of BVH node numbers.*

Precision		10^{-3}		10^{-4}		10^{-5}
Models	PQP	Ours	PQP	Ours	PQP	Ours
pawn	287K	103	2.5M	211	26.5M	479
cup	272K	119	3.0M	199	25.9M	399
bottle	308K	123	2.7M	187	29.2M	411
table	289K	95	3.1M	179	26.3M	335
lightmil	424K	247	4.5M	375	42.5M	687

volumes with cubic convergence to planar curves [Barton and Elber (2011), Kumosenko (2013)].)

For each pair of static solid models in Figure 7, we have also computed the minimum distances between the two models at 50,000 different frames taken from their continuous motion. Figure 11 shows some snapshots of the test for the pairs of (table, lightmil) and (pawn, bottle) models. Figure 12 reports a total of 15 test results thus computed. The height of each vertical bar gives the average computation time for the 50,000 distances for each pair. Depending on the applications of minimum distance computation, we need different levels of precision in the model representation and accuracy in the computed result. Three different values of $\varepsilon = 10^{-3}, 10^{-4}, 10^{-5}$ are used in these tests. The difference in computation time is also partially affected by the relative sizes (and the heights) of the BVH structures as shown in Tables 1–2.

6. Conclusions

In this paper, we have presented an efficient algorithm for computing the minimum distance between two solids of revolution, which can support real-time applications for several deformable models at the same time by generating their BVH structures on the fly. The efficiency improvement is not only in computation time (often 10–100 times faster than other conventional methods) but also in data size (often 1000 times smaller). The idea of our approach is mainly based on the geometric simplicity of circles and the high-order (cubic) convergence of G^1-biarc approximation. Nevertheless, the geometric coverage of the current result is still limited to solid models of revolution. In future work, we plan to extend the coverage to more general types of freeform geometric models and to other efficient geometric algorithms and spatial data structures for distance-related problems.

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-2018R1D1A1B07048036, NRF-2019R1A2C1003490, NRF-2019K1A3A1A78112596), and in part by the ISRAEL SCIENCE FOUNDATION (grant No. 597/18).

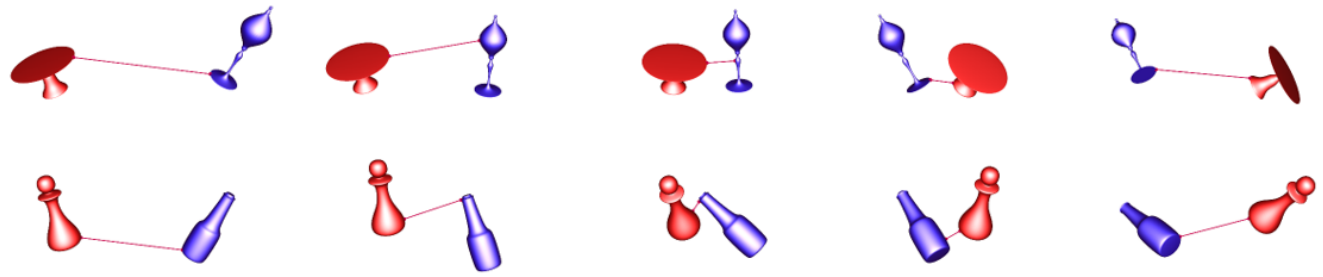

Figure 11: *Snapshots from the test for the pairs of table-lightmil and pawn-bottle models.*

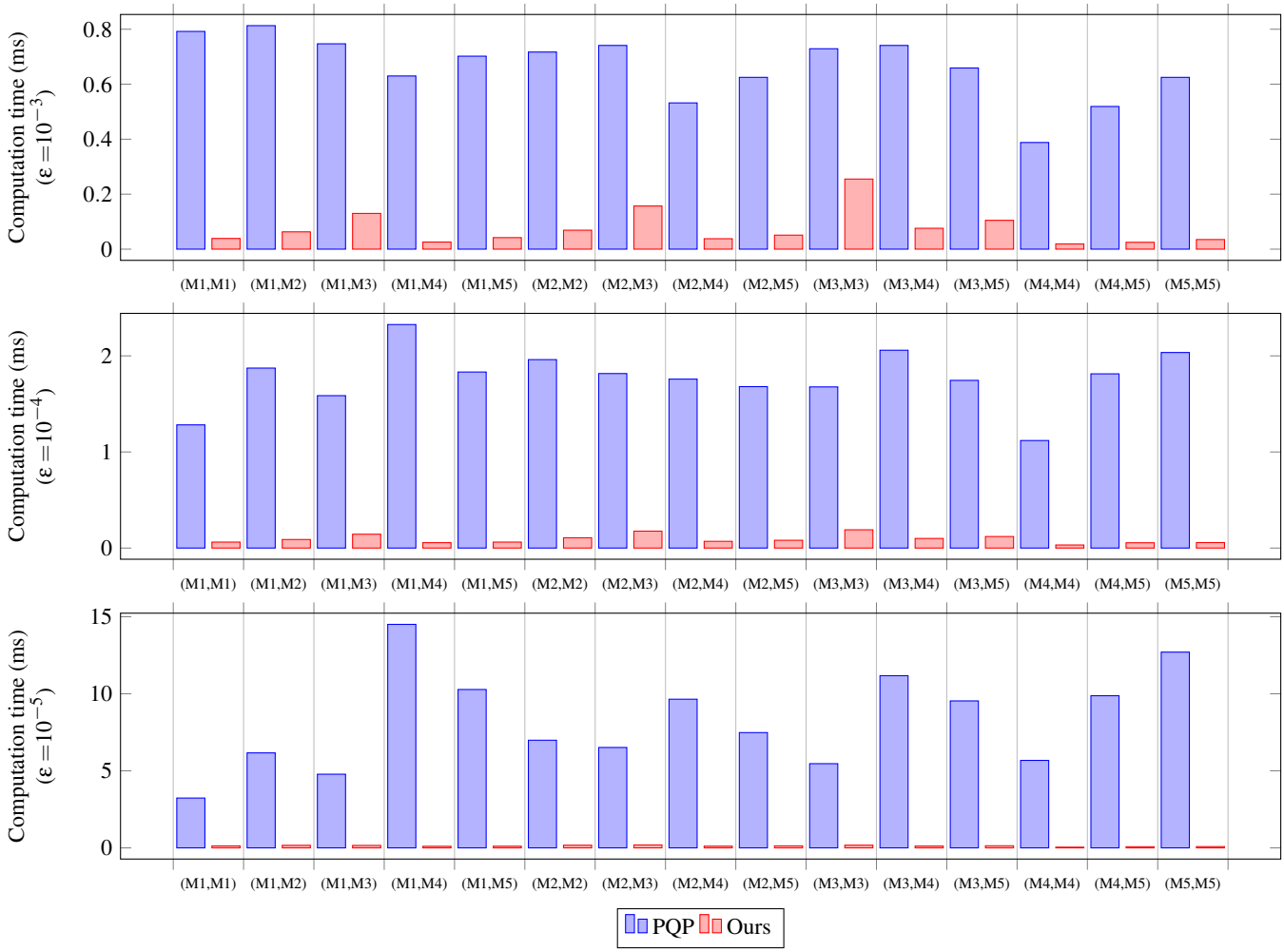

Figure 12: *Comparison of the relative performance of our algorithm against the approach of [Larsen et al. 1999].*

References

[Barton and Elber (2011)] M. Barton and G. Elber: Spiral fat arcs – Bounding regions with cubic convergence. *Graphical Models*, **73**(2):50–57, 2011. 7

[Bo and Barton (2019)] P. Bo and M. Barton: On initialization of milling paths for 5-axis flank CNC machining of free-form surfaces with general milling tools. *Computer Aided Geometric Design*, **71**:30–42, 2019. 2

[Cameron (1997)] S. Cameron: Enhancing GJK: computing minimum and penetration distances between convex polyhedra. *Proc. of the 1997 IEEE Int'l Conf. on Robotics and Automation*,, pp. 3112–3117, Albuquerque, New Mexico, April 20–25, 1997. 3

[Chang (2011)] J.-W. Chang, Y.-K. Choi, M.-S. Kim, and W. Wang: Computation of the minimum distance between two Bézier curves/surfaces. *Computers & Graphics*, **35**(3):677–684, 2011. 3

[Eberley (2007)] D.H. Eberley: *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*, 2nd Ed., Morgan Kaufmann, San Francisco, 2007. 3

[Gilbert et al. (1988)] E. Gilbert, D. Johnson, S. Keerthi: A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Journal of Robotics and Automation **4**(2):193–203, 1988. 1, 3

[Gottschalk et al. (1996)] S. Gottschalk, M. Lin, D. Manocha: OBB-Tree: A hierarchical structure for rapid interference detection. Proc. of SIGGRAPH 96, 171–180, 1996. 6

[Johnson (2005)] D. Johnson: Minimum distance queries for haptic rendering. PhD thesis, Computer Science Department, University of Utah, 2005. 3

[Johnson and Cohen (1998)] D. Johnson, E. Cohen: A framework for efficient minimum distance computations. *IEEE Int'l Conf. on Robotics and Automation*, pp. 3678–3684, 1998. 3

[Kim et al. (2011)] Y.-J. Kim, Y.-T. Oh, S.-H. Yoon, M.-S. Kim, and G. Elber: Coons BVH for freeform geometric models. *ACM Transactions on Graphics*, **30**(6): Article 169, SIGGRAPH Asia 2011. 3

[Kim et al. (2012)] Y.-J. Kim, J. Lee, M.-S. Kim, and G. Elber: Efficient offset trimming for planar rational curves using biarc trees. *Computer Aided Geometric Design*, **29**(7):555–564, 2012. 3

[Krishnan et al. 1998a] S. Krishnan, M. Gopi, M. Lin, D. Manocha, S. Pattekar: Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum*, **17**(3):315–326, 1998. 1, 3

[Kumosenko (2013)] A. Kumosenko. Biarcs and bilens, *Computer Aided Geometric Design*, **30**(3):310–330, 2013. 5, 7

[Larsen et al. 1999] E. Larsen, S. Gottschalk, M. Lin, D. Manocha: Fast proximity queries using swept sphere volumes. Technical Report TR99-018, Dept. of Computer Science, UNC, 1999. 3, 5, 6, 8

[Lee et al. (2015a)] J. Lee, Y.-J. Kim, M.-S. KIm, and G. Elber: Efficient offset trimming for deformable planar curves using a dynamic hierarchy of bounding circular arcs, *Computer-Aided Design*, **58**:248–255, 2015. 1, 3

[Lee et al. (2015b)] J. Lee, Y.-J. Kim, M.-S. KIm, and G. Elber: Comparison of three bounding regions with cubic convergence to planar freeform curves. *The Visual Computer*, **31**(6-8):809–818, 2015. 6

[Lee et al. (2016)] J. Lee, Y.-J. Kim, M.-S. KIm, and G. Elber: Efficient Voronoi diagram construction for planar freeform spiral curves. *Computer Aided Geometric Design*, **43**:131–142, 2016. 1, 3

[Lin and Canny (1991)] M. Lin, J. Canny: A fast algorithm for incremental distance calculation. In: Proc. of IEEE International Conference on Robotics and Automation. pp. 1008–1014, 1991. 1, 3

[Lin and Manocha (2004)] M. Lin, D. Manocha: Collision and proximity queries. *Handbook of Discrete and Computational Geometry*, 2nd Ed., J.E. Goodman and J. O'Rourke, Eds., Chapman & Hall/CRC, pp. 787–807, 2004. 3

[Lin and Rokne (2002)] Q. Lin and J.G. Rokne: Approximation by fat arcs and fat biarcs. *Computer-Aided Design*, **34**(13):969–979, 2002. 3

[Liu et al. (2009)] X.-M. Liu, L. Yang, J.-H. Yong, H.-J. Gu, J.-G. Sun: A torus patch approximation approach for point projection on surfaces, *Computer Aided Geometric Design*, **26**(5):593–598, 2009 2, 3

[Machchhar and Elber (2016)] J. Machchhar and G. Elber: Revisiting the problem of zeros of univariate scalar Bźiers. *Computer Aided Geometric Design*, **43**:16–26, 2016. 4

[Meek and Walton (1995)] D. Meek and D. Walton: Approximating smooth planar curves by arc splines. *J. of Computational and Applied Mathematics*, **59**(2):221–231, 1995. 3

[Neff (1990)] C.A. Neff: Finding the distance between two circles in three-dimensional space. *IBM J. of Research and Development*, **34**(5):770–775, 1990. 3

[Quinlan (1994)] S. Quinlan. Efficient distance computation between non-convex objects. *IEEE Int'l Conf. on Robotics and Automation*, pp. 3324–3329, 1994. 1

[Sederberg et al. (1989)] T.W. Sederberg, S.C. White, and A.K. Zundel: Fat arcs: A bounding region with cubic convergence. *Computer Aided Geometric Design*, **6**(3):205–218, 1989. 1, 3

[Sir et al. (2006)] Z. Sir, R. Feichtinger, B. Juttler: Approximating curves and their offsets using biarcs and Pythagorean hodograph quintics. *Computer-Aided Design*, **38**(6):608–618, 2006. 2, 5

[van den Bergen (1999)] G. van den Bergen: A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, **4**(2):7–25, 1999. 3

[Vranek (2002)] D. Vranek. Fast and accurate circle-circle and circle-line 3D distance computation. *J. of Graphics Tools*, **7**(1):23–32, 2002. 3

[Yong et al. (2006)] J.-H. Yong, X. Chen, J.-C. Paul: An example on approximation by fat arcs and fat biarcs. *Computer-Aided Design*, **38**(5):515–517, 2006. 3

Appendix A: Derivation of Equations for Binormal Conditions

The coplanarity condition for three vectors: $C_1(\theta_1)$, $C_2(\theta_2)$, and $\mathbf{e}_3$, implies that their determinant must vanish:

$$
\begin{aligned}
& |C_1(\theta_1)\; C_2(\theta_2)\; \mathbf{e}_3| \\
= & \begin{vmatrix} R_1\cos\theta_1 & R_2(u_x\cos\theta_2 + v_x\sin\theta_2) + t_x & 0 \\ R_1\sin\theta_1 & R_2(u_y\cos\theta_2 + v_y\sin\theta_2) + t_y & 0 \\ 0 & R_2(u_z\cos\theta_2 + v_z\sin\theta_2) + t_z & 1 \end{vmatrix} \\
= & \begin{vmatrix} R_1\cos\theta_1 & R_2(u_x\cos\theta_2 + v_x\sin\theta_2) + t_x \\ R_1\sin\theta_1 & R_2(u_y\cos\theta_2 + v_y\sin\theta_2) + t_y \end{vmatrix} = 0,
\end{aligned}
$$

where $\mathbf{u} = (u_x, u_y, u_z)$ and similarly for $\mathbf{v}$ and $\mathbf{t}$. This equation can be simplified as follows

$$[R_2(u_y\alpha_2 + v_y\beta_2) + t_y]\alpha_1 - [R_2(u_x\alpha_2 + v_x\beta_2) + t_x]\beta_1 = 0,$$

where $\alpha_i = \cos\theta_i$ and $\beta_i = \sin\theta_i$, for $i = 1, 2$.

Switching the roles of T_1 and T_2, in the local coordinates of T_2, the major circle of T_1 can be represented as follows:

$$\hat{C}_1(\theta_1) = \hat{\mathbf{u}}\, R_1\cos\theta_1 + \hat{\mathbf{v}}\, R_1\sin\theta_1 + \hat{\mathbf{t}},$$

which produces the following equation

$$[R_1(\hat{u}_y\alpha_1 + \hat{v}_y\beta_1) + \hat{t}_y]\alpha_2 - [R_1(\hat{u}_x\alpha_1 + \hat{v}_x\beta_1) + \hat{t}_x]\beta_2 = 0,$$

and equivalently

$$R_1(\hat{u}_y\alpha_2 - \hat{u}_x\beta_2)\alpha_1 + R_1(\hat{v}_y\alpha_2 - \hat{v}_x\beta_2)\beta_1 = \hat{t}_x\beta_2 - \hat{t}_y\alpha_2.$$

Summarizing the above in a matrix equation,

$$\begin{bmatrix} R_2(u_y\alpha_2+v_y\beta_2)+t_y & -[R_2(u_x\alpha_2+v_x\beta_2)+t_x] \\ R_1(\hat{u}_y\alpha_2-\hat{u}_x\beta_2) & R_1(\hat{v}_y\alpha_2-\hat{v}_x\beta_2) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} 0 \\ \hat{t}_x\beta_2-\hat{t}_y\alpha_2 \end{bmatrix}$$

By Cramer's rule, we have

$$\alpha_1 = \frac{[\hat{t}_x\beta_2-\hat{t}_y\alpha_2][R_2(u_x\alpha_2+v_x\beta_2)+t_x]}{Q(\alpha_2,\beta_2)}$$
$$\beta_1 = \frac{[\hat{t}_x\beta_2-\hat{t}_y\alpha_2][R_2(u_y\alpha_2+v_y\beta_2)+t_y]}{Q(\alpha_2,\beta_2)},$$

where $Q(\alpha_2,\beta_2)$ is a quadratic polynomial in α_2 and β_2:

$$\begin{aligned} Q(\alpha_2,\beta_2) = {} & R_1[R_2(u_y\alpha_2+v_y\beta_2)+t_y](\hat{v}_y\alpha_2-\hat{v}_x\beta_2) \\ & + R_1[R_2(u_x\alpha_2+v_x\beta_2)+t_x](\hat{u}_y\alpha_2-\hat{u}_x\beta_2). \end{aligned}$$

The condition $\alpha_1^2+\beta_1^2=1$ can be transformed to the following quartic polynomial equations $P(\alpha_2,\beta_2)=0$ in α_2 and β_2:

$$\begin{aligned} P(\alpha_2,\beta_2) = {} & [\hat{t}_x\beta_2-\hat{t}_y\alpha_2]^2[R_2(u_x\alpha_2+v_x\beta_2)+t_x]^2 \\ & + [\hat{t}_x\beta_2-\hat{t}_y\alpha_2]^2[R_2(u_y\alpha_2+v_y\beta_2)+t_y]^2 \\ & - Q(\alpha_2,\beta_2)^2. \end{aligned}$$

From the two simultaneous equations $P(\alpha_2,\beta_2)=0$ and $\alpha_2^2+\beta_2^2=1$, by eliminating β_2, we get a degree 8 polynomial equation in α_2. This shows that there can be at most 8 binormal lines to two circles in a three-dimensional space.

We can show the elimination procedure more explicitly. From the condition $\alpha_2^2+\beta_2^2=1$, we have $\beta_2=\pm\sqrt{1-\alpha_2^2}$. The bivariate polynomial equation $P(\alpha_2,\beta_2)=0$ can be reduced to a univariate non-polynomial equation in α_2, using one square root expression:

$$P(\alpha_2,\pm\sqrt{1-\alpha_2^2}) = A(\alpha_2)\pm B(\alpha_2)\sqrt{1-\alpha_2^2}=0,$$

where $A(\alpha_2)$ and $B(\alpha_2)$ are univariate polynomials of degree 4 and 3, respectively. The above equation can be converted to a univariate polynomial equation of degree 8 in α_2:

$$A(\alpha_2)^2-(1-\alpha_2^2)B(\alpha_2)^2=0.$$