Precise Hausdorff Distance Computation for Freeform Surfaces Based on Computations with Osculating Toroidal Patches

Sang-Hyun Son[a], Myung-Soo Kim[a], Gershon Elber[b]

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

Abstract

We present an efficient algorithm for computing the precise Hausdorff Distance (HD) between two freeform surfaces. The algorithm is based on a hybrid Bounding Volume Hierarchy (BVH), where osculating toroidal patches (stored in the leaf nodes) provide geometric properties essential for the HD computation in high precision. Intrinsic features from the osculating geometry resolve computational issues in handling the cross-boundary problem for composite surfaces, which leads to the acceleration of HD algorithm with a solution (within machine precision) to the exact HD. The HD computation for general freeform surfaces is discussed, where we focus on the computational issues in handling the local geometry across surface boundaries or around surface corners that appear as the result of gluing multiple patches together in the modeling of generic composite surfaces. We also discuss how to switch from an approximation stage to the final step of computing the precise HD using numerical improvements and confirming the correctness of the HD computation result. The main advantage of our algorithm is in the high precision of HD computation result. As the best cases of the proposed torus-based approach, we also consider the acceleration of HD computation for freeform surfaces of revolution and linear extrusion, where we can support real-time computation even for deformable surfaces. The acceleration is mainly due to a fast biarc approximation to the planar profile curves of the simple surfaces, each generated by rotating or translating a planar curve. We demonstrate the effectiveness of the proposed approach using experimental results.

Keywords: Hausdorff Distance, freeform surfaces, Bounding Volume Hierarchy (BVH), osculating toroidal patches, intrinsic geometry, precise computation, cross-boundary problem, surface of revolution, surface of linear extrusion

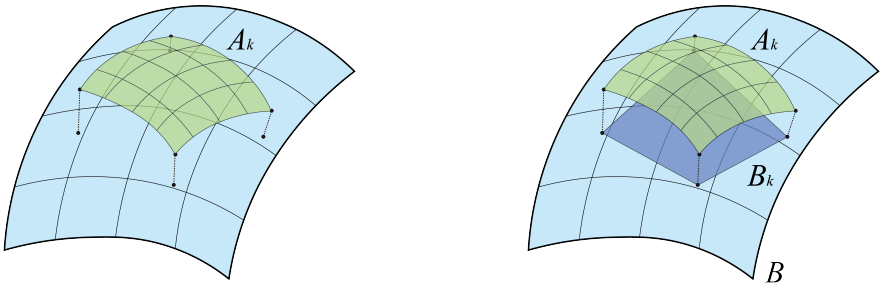
1. Introduction

Given two objects A and B, the maximum deviation of A from the other object B can be measured using the one-sided Hausdorff Distance (HD) of A from B:

$$h(A,B) = \max_{\mathbf{p}\in A}\left(\min_{\mathbf{q}\in B}\|\mathbf{p}-\mathbf{q}\|\right) = \max_{\mathbf{p}\in A} d(\mathbf{p},B),$$

where $d(\mathbf{p},B)$ is the minimum distance of $\mathbf{p}$ from B. The one-sided HD being zero ($h(A,B)=0$) means that an object A is completely contained in the other object B. In other words, $h(A,B)=0$ implies $A\subset B$, and vice versa. The identity of two objects $A\equiv B$ can be detected computationally by measuring $h(A,B)=0$ and $h(B,A)=0$. On the other hand, the similarity of two shapes $A\approx B$ can be represented quantitatively using two conditions: $h(A,B)<\epsilon$ and $h(B,A)<\epsilon$, for a small value $\epsilon>0$, which can be set appropriately by the user, depending on each application of the similarity test under consideration.

*Corresponding author
Email address: mskim@snu.ac.kr (Myung-Soo Kim)



(a) (b)

Figure 1: Matching two nearby surface patches: (a) the four corners of A_k are projected to the other surface B, and (b) a surface patch B_k of similar size is extracted from B by reparameterizing $B_k(u,v) = B(s(u,v), t(u,v))$.

The (two-sided) Hausdorff Distance (HD) between A and B is defined as the maximum of the two one-sided HDs measured from both sides:

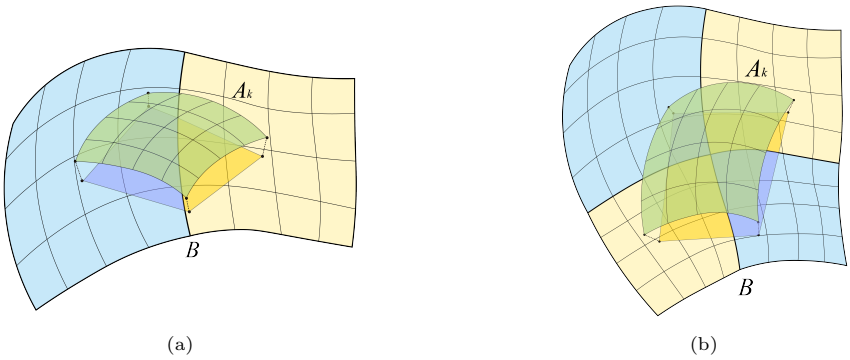
$$H(A,B) = \max\left(h(A,B), h(B,A)\right).$$

The identity $A \equiv B$ is thus equivalent to their HD being zero: $H(A,B) = 0$. Based on this relation, the HD computation has been extensively used as an important tool for shape matching and recognition [1; 22]. For the sake of convenience, by changing the roles of A and B if necessary, we may assume $h(A,B) \geq h(B,A)$ and thus $H(A,B) = h(A,B)$ in the rest of this paper.

The majority of HD computation algorithms try to make a good guess on the maximum deviation point $\mathbf{p}^* \in A$ from B: $d(\mathbf{p}^*, B) = h(A,B)$, where $d(\mathbf{p}^*, B)$ is the distance between $\mathbf{p}^*$ and B. To speed up the search for the solution point $\mathbf{p}^*$, we need to reduce the search space from the whole object A to a much smaller subset $A^* \subset A$, often by removing redundant parts of A from further consideration. For two parametric freeform surfaces $A(u,v)$ and $B(s,t)$, Kim et al. [16] introduced a *matching* technique for this purpose, the basic concept of which was also used in the recent results of Kang et al. [13; 14] for computing the HD between triangular and quad meshes. The main advantage of Kang et al. [13; 14] is the guarantee of high precision in the HD computation result, where the computing time for a tight error bound 10^{-9} takes only slightly more than that for a larger error bound 10^{-3}. In the current work, we propose a new approach to the problem for freeform surfaces, which can significantly improve the precision of HD computation compared with the previous results on freeform surfaces [16]. Below we discuss some difficulties expected in this new approach.

As illustrated in Figure 1, using a local 1-to-1 matching between a subpatch $A_k(u,v)$ of A and another subpatch $B_k(u,v)$ of B, Kim et al. [16] introduced an effective technique for estimating an upper bound $\overline{h}_k$ for $h(A_k, B)$, the one-sided HD from A_k to B. When the upper bound $\overline{h}_k$ is smaller than the distance $d(\mathbf{p}, B)$, for some point $\mathbf{p} \in A$, the subpatch A_k is redundant from the computation of $h(A,B)$. This is because none of A_k may contain a point of larger deviation (from B) than the distance $d(\mathbf{p}, B)$; thus, in A_k, we cannot find a maximum deviation point $\mathbf{p}^*$ of A (from B), i.e., a point $\mathbf{p}^* \in A$ such that $d(\mathbf{p}^*, B) = h(A,B)$. The subpatch A_k is removed from the HD computation, which is the main source of acceleration in the matching-based approach to the HD computation. The reparameterization $B_k(u,v) = B(s(u,v), t(u,v))$ is relatively easy when the matching subpatch B_k is totally contained in the same Bézier surface patch of a composite surface $B(s,t)$. (In Section 3, we briefly review a simple technique for estimating an upper bound $\overline{h}_k = \max \|\mathbf{d}_{ij}\|$ using a difference surface $D(u,v) = A_k(u,v) - B_k(u,v)$ with control points $\{\mathbf{d}_{ij}\}$.)

Figure 2 shows two non-trivial cases where an ideal matching would be made across a common boundary of two subpatches or around a corner point shared among four subpatches. Even when the subpatches are connected smoothly with a sufficiently high C^k-continuity, they may have different rational/polynomial representations. Ideally speaking, the subpatch A_k could be subdivided into smaller pieces of similar shape to the corresponding counterparts on the surface B. In practice, this can be done efficiently for simple surfaces such as triangular or quad meshes. (In some sense, Kang et al. [13; 14] follow the basic guideline



(a) (b)

Figure 2: Matching across surface boundaries: (a) the four corners of A_k are projected to two different subpatches of B, and (b) to four different subpatches of B sharing a common corner point.

of this idea.) Nevertheless, for general composite B-spline surfaces, it is highly non-trivial to develop an efficient HD algorithm by simply subdividing into subpatches of arbitrary shape. There are some technical issues that must be resolved, which we discuss below, for the development of an efficient HD algorithm that can also guarantee a high-precision accuracy in the HD computation result.

Kim et al. [16] dealt with the cases of Figure 2 by recursively subdividing the subpatch A_k into smaller rectangular pieces, which often slows down the algorithm, mainly due to the fact that the subdivisions are made along the u and v-directions of $A_k(u, v)$ and the cross-boundary problem of Figure 2 is repeated to a large number of smaller pieces thus subdivided. (Subdivision of $A_k(u, v)$ along other directions is more expensive.) In the test examples of Kim et al. [16], the error bound for HD approximation was set to 10^{-3}, where the unit length 1 is taken as the model size. (The error bounds of $10^{-2}, 10^{-3}, 10^{-4}$ are often used in many HD algorithms based on iterative approximations [16; 17; 25].) On the other hand, for polygonal meshes, Kang et al. [13; 14] made considerable improvement in the precision of HD computation, up to an error bound 10^{-9}, using only slightly (1–2%) more computing time than the case of error bound 10^{-3}. To get a similar improvement in the HD computation for freeform surfaces, we may need to support subdivisions of A_k in arbitrary directions. Nevertheless, this approach has drawbacks in some non-trivial cases.

Freeform geometric models often contain polar points such as those appearing in the north/south poles of a sphere. Several freeform surfaces meet at a polar point by collapsing their edges to the polar point. When the maximum deviation HD point is located near the polar point, the HD computation can be slowed down quite significantly. To deal with this problem, we use toroidal patches tightly fitting the neighborhood of a polar point, which has produced an efficient and stable HD computation result. This approach also works for handling the cross-boundary problem for adjacent surface patches with different equations. The main advantage of using toroidal patches is in computing the local and global minimum/maximum distances between two toroidal patches. Son et al. [24] demonstrated the power of torus-based geometric approach by accelerating the minimum distance computation for solids of revolution. In this paper, we show that a similar approach can be applied to the HD computation for surfaces of revolution and linear extrusion, and further to general freeform surfaces.

Our HD algorithm is based on two hybrid BVHs for $A(u, v)$ and $B(s, t)$, where their leaf nodes contain osculating toroidal patches with second-order contact with these surfaces at their mid-parameters. At the leaf level of our hybrid BVH, the comparison between two subpatches $A_k(u, v)$ and $B_k(u, v)$ is replaced by an approximate comparison between their osculating toroidal patches $T_A(u, v)$ and $T_B(u, v)$, which is simpler and more efficient to deal with than the general surfaces. For the sake of efficiency in BVH construction, we use the second order osculating torus patch of Liu et al. [20], originally developed for the acceleration of point-projection algorithm. (The HD computation is essentially the coordination of many point-projections.) Using a boundary trimming technique illustrated in Figure 4 of Park et al. [21], we generate the osculating toroidal patches (properly reparameterized) so that, for each (u, v), the difference in positions $A_k(u, v)$ and $T_A(u, v)$ (respectively, $B_k(u, v)$ and $T_B(u, v)$) is smaller than a given tolerance.

For deformable geometric models, the BVH structure should be updated on the fly and the update time is included in the HD computing time. We demonstrate a real-time HD computation (including the BVH update) for deformable surfaces of revolution and linear extrusion (generated by rotating and translating a deformable planar profile curve), which is considerably faster than the computation for general freeform surfaces. The acceleration on these special cases is based on an efficient construction algorithm for the biarc trees for planar profile curves of these surfaces.

Once we have approximated the HD within an error bound 10^{-9} (by dynamically subdividing the subpatches in the leaf levels if necessary), we then switch to the final stage of computing a precise HD within machine precision to the exact HD. The precise HD computation of Barton et al. [5] reports that a considerable slow down of the HD algorithm may occur when the maximum deviation point $\mathbf{p}^* \in A$ is located on the self-bisector surface of B, which means that $\mathbf{p}^*$ has two different footpoints $\mathbf{q}_1, \mathbf{q}_2 \in B$. The fast (but approximate) HD algorithm of Tang et al. [25] does not consider the bisector case by repeatedly subdividing the neighborhoods of $\mathbf{p}^*, \mathbf{q}_1, \mathbf{q}_2$ into smaller and smaller pieces. When applied to freeform surfaces, this approach would produce large errors in (representing the control points for) recursively subdivided subpatches of the input surfaces. Moreover, the surface subdivisions are considerably more time-consuming than the case of polygonal meshes. We switch to the algorithm of Elber and Grandine [8], where the precise HD is computed by solving a system of multivariate polynomial equations.

Our main contributions can be summarized as follows:

- We present an efficient algorithm for computing the HD of two freeform surfaces, within machine precision to the exact HD.
- For surfaces of special types such as surfaces of revolution and linear extrusion, our algorithm supports a real-time HD computation for the freeform surfaces under continuous deformation.
- Using osculating toroidal patches that approximate the given freeform surfaces very tightly, we have greatly improved the precision of HD computation to a similar level to the results of Kang et al. [13; 14] for polygonal meshes, and even to the ultimate level (of machine precision) reported in the previous result of Elber and Grandine [8] for freeform surfaces.
- We introduce a torus-based technique that can effectively handle the cross-boundary as well as polar point problems in the HD computation.

The rest of this paper is organized as follows. In Section 2, we review previous research results on the HD computation and related problems. Section 3 briefly reviews basic geometric concepts and discusses some technical issues and practical solutions. Our BVH-based algorithm is then explained in Section 4, where we focus on the main differences from the previous methods. In Section 5, we demonstrate the performance of our HD computation algorithm using several experimental results. Finally, in Section 6, we conclude this paper.

2. Related Work

The HD computation for polygonal meshes has a long history with many fruitful research results [3; 7; 10; 13; 14; 25]. Introducing a new computational paradigm based on the spatial data structure of Bounding Volume Hierarchy (BVH), Tang et al. [25] developed the first interactive-speed HD algorithm for polygonal meshes. Barton et al. [5] considered the precise HD computation for polygonal meshes. The recent work of Kang et al. [13; 14] demonstrated that the HD between triangular and quad meshes can be computed up to a high precision such as within an error bound of 10^{-9} in about the same computing time as in the low precision cases. The basic approach of Kang et al. [13; 14] can be extended to the more general case of computing the HD between freeform surfaces. Nevertheless, it is a great challenge to develop a precise HD algorithm that works in an interactive speed for reasonably complex geometric models composed of freeform composite surfaces, which we attack in the current work.

Elber and Grandine [8] computed the precise HD between two freeform rational surfaces by solving a system of polynomial constraint equations. By restricting the domain of parameters (by applying a new

pruning technique), we accelerate the computing speed while keeping the high precision final result, which is the approach we take in the current work. The acceleration technique is based on osculating toroidal patches to the freeform surfaces under consideration. The osculating torus has been shown to be quite effective in accelerating geometric problems related to the minimum distance of freeform surfaces [20; 21; 24]. This paper demonstrates the effectiveness even in the maximum distance computation.

Using GPU acceleration techniques, Krishnamurthy et al. [17] introduced a real-time HD algorithm for dynamically deformable NURBS surfaces, and Hanniel et al. [11] developed an HD algorithm that can handle the non-trivial cases of near-overlap and near-offset surfaces. On the other hand, Kim et al. [16] presented a purely CPU-based real-time HD algorithm for NURBS surfaces, which can also deal with the near-overlap and near-offset cases successfully. The approach of Kim et al. [16] is based on a BVH structure, specially designed for NURBS surfaces. In this paper, we modify the BVH structure by adding osculating toroidal patches to the leaf nodes and improve the performance of redundancy pruning by developing efficient torus-based techniques for bounding local HDs for surface patches.

There are also many HD computation results for freeform curves. Jüttler [12] estimated upper bounds for the HD between two freeform/implicit curves. Alt and Scharf [2] considered the precise HD computation for planar rational curves. Using graphics hardware depth buffer, Kim et al. [15] computed the precise HD for planar freeform curves in real-time. The HD algorithms of Chen et al. [6] and Bai et al. [4] are also based on pruning techniques for redundant segments from the given freeform curves.

The surfaces of revolution or linear extrusion are generated by rotating or translating planar profile curves. Son et al. [24] presented a highly effective BVH structure for the surfaces of revolution using a G^1-biarc tree for planar profile curves. There is one degree of freedom in the selection of each biarc [23], which has promoted the development of many different types of biarc. In particular, Kurnosenko [18] introduced a bounding region, called bilens, based on osculating circles to planar curves. By rotating (or translating) the bilens and osculating circles, we can generate bounding volumes and osculating tori (or osculating cylinders) for the surfaces of revolution (or linear extrusion). Lee et al. [19] made detailed comparison among arc-based BVH trees for planar freeform curves.

3. Basic Concepts and Geometric Tools

As illustrated in Figure 1, using a local 1-to-1 matching between a subpatch $A_k(u,v)$ of A and another subpatch $B_k(u,v)$ of B, reparameterized as

$$B_k(u,v) = B(s(u,v), t(u,v)),$$

the matching technique of Kim et al. [16] provides an effective way of estimating an upper bound for $h(A_k, B)$, the one-sided HD from A_k to B. We briefly review how to compute an upper bound $\overline{h}_k$ for the one-sided Hausdorff distance $h(A_k, B)$, and discuss some technical issues related to the cross-boundary problem. Moreover, we discuss how to simplify the test $h(A_k, B) < \underline{h}$, using an osculating toroidal patch of B_k, which can be used for the elimination of A_k from further consideration.

3.1. Matching-based upper bound for local HD

Note the following simple relationship:

$$\|A_k(u,v) - B_k(u,v)\| \geq d(A_k(u,v), B),$$

where $d(A_k(u,v), B)$ is the minimum distance from a surface point $A_k(u,v)$ to the other surface B. Taking maximum from both sides, we have

$$\max_{(u,v)} \|A_k(u,v) - B_k(u,v)\| \geq \max_{(u,v)} d(A_k(u,v), B) (= h(A_k, B)).$$

Representing the difference $D(u,v) = A_k(u,v) - B_k(u,v)$ as a freeform surface with control points $\{\mathbf{d}_{ij}\}$,

$$A_k(u,v) - B_k(u,v) = \sum\sum \mathbf{d}_{ij} B_i(u) B_j(v),$$

we can bound $h(A_k, B)$ and $\max \|A_k(u,v) - B_k(u,v)\|$ by $\max \|\mathbf{d}_{ij}\|$ as follows

$$\|A_k(u,v) - B_k(u,v)\| = \left\| \sum \sum \mathbf{d}_{ij} B_i(u) B_j(v) \right\| \leq \sum \sum \|\mathbf{d}_{ij}\| B_i(u) B_j(v) \leq \max \|\mathbf{d}_{ij}\|,$$

and thus

$$h(A_k, B) \leq \max \|A_k(u,v) - B_k(u,v)\| \leq \max \|\mathbf{d}_{ij}\|.$$

We can take $\overline{h}_k = \max \|\mathbf{d}_{ij}\|$ as an upper bound for $h(A_k, B)$.

In the reparameterization of $B_k(u,v) = B(s(u,v), t(u,v))$, when we use bilinear functions $s(u,v), t(u,v)$, the degree of $B_k(u,v)$ will be doubled due to the uv-term of the bilinear functions. For a bicubic surface, the number of control points will be almost tripled, the construction of which also takes quite some time in the symbolic process for the composition of spline functions. A more efficient way of estimating an upper bound $\overline{h}_k$ is to take the maximum of $\|A_k(u_a, v_b) - B_k(u_a, v_b)\|$ computed at uniform samples of (u_a, v_b) and add an error term due to the uniform sampling of uv-parameters (which can be computed by the condition of Filip et al. [9]).

3.2. Some technical issues related to the cross-boundary problem

In the above discussion, we have implicitly assumed that the range of $(s(u,v), t(u,v))$ is totally contained in the domain of a Bézier surface $B(s,t)$, with a well-defined polynomial/rational surface equation and a set of control points. When the four corner points of A_k are projected to $B(s_{ij}, t_{ij})$, $(i,j = 0,1)$, with all four solutions $(s_{ij}, t_{ij}) \in [0,1] \times [0,1]$, in the domain of a Bézier surface $B(s,t)$, we can bilinearly reparameterize the quadrangular region Q formed by these four (s_{ij}, t_{ij})-solutions as follows:

$$Q(u,v) = (s(u,v), t(u,v)) = (1-u)(1-v)\ (s_{00}, t_{00}) + u(1-v)\ (s_{10}, t_{10}) + (1-u)v\ (s_{01}, t_{01}) + uv\ (s_{11}, t_{11}).$$

When the quadrangle Q forms a parallelogram or even a rectangle, the uv-term vanishes and we have a linear reparameterization of $(s(u,v), t(u,v))$:

$$Q(u,v) = (s(u,v), t(u,v)) = (s_{00}, t_{00}) + u\ (s_{10} - s_{00}, t_{10} - t_{00}) + v\ (s_{01} - s_{00}, t_{01} - t_{00}),$$

which produces $B_k(u,v) = B(s(u,v), t(u,v))$ of the same degree as $B(s,t)$. In general, due to the uv-term, the degree of $B_k(u,v)$ is twice as higher as that of $B(s,t)$.

When the projections of four corner points are made to two adjacent Bézier surfaces $B_1(s,t)$ and $B_2(s,t)$, which form a composite surface $B(s,t), ((s,t) \in [0,2] \times [0,1])$, for example, $B_1(s,t), ((s,t) \in [0,1] \times [0,1])$, and $B_2(s,t), ((s,t) \in [1,2] \times [0,1])$, the vertical line $s = 1$ splits the quadrangle $Q(u,v)$ into two pieces $Q_1(u,v)$ and $Q_2(u,v)$. Note that, because of the uv-term in the bilinear reparameterization, in the uv-parameter domain, the split will be along an implicit quadratic curve:

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

which makes it difficult to represent $B(Q_1(u,v))$ and $B(Q_2(u,v))$ in a common functional space. (This means that we may not find the control points for the difference surface $D(u,v) = A_k(u,v) - B_k(u,v)$.) Nevertheless, we can evaluate $\|A_k(u_a, v_b) - B_k(u_a, v_b)\|$ on uniform samples (u_a, v_b), using either $B_1(s,t)$ or $B_2(s,t)$, depending on the sign of $s(u_a, v_b) - 1$. When the two Bézier surfaces $B_1(s,t)$ and $B_2(s,t)$ are connected with C^2-continuity, we can apply the result of Filip et al. [9] to the uniform sampling technique and estimate an upper bound for the maximum difference. On the other hand, when they are only C^1-continuous, we cannot use the C^2-based formulas of Filip et al. [9] for bounding the influence of sampling errors. Moreover, when there is even no guarantee of G^1 or C^0-continuity, we need to consider the boundary curve segments separately from the surface interiors. Boundary curves, corner points, and cusps may be considered as degenerate surfaces.

In all these cases of non-C^2-continuity, we may reparameterize the two pieces Q_1 and Q_2 separately, and split A_k into two pieces A_{k_1} and A_{k_2} of similar shapes to Q_1 and Q_2. (As the result of split, Q_1 or Q_2 may also be a triangle.) The subdivision of A_k along a direction other than the u or v-direction raises the degrees of A_{k_1} and A_{k_2}. All these computational hurdles motivate the introduction of osculating toroidal patches, as a handy tool for replacing the surface matching and splitting by simpler tasks on toroidal patches.

3.3. Torus-based pruning

We interpret the condition $h(A_k, B_k) < \epsilon$ as the containment of A_k in the offset volume of B_k by $\epsilon > 0$:

$$A_k \subset O_\epsilon(B_k) = \{\ \mathbf{p} \mid \|\mathbf{p} - \mathbf{q}\| < \epsilon,\ \mathbf{q} \in B_k\} = \cup_{\mathbf{q} \in B_k}\ O_\epsilon(\mathbf{q}),$$

where $O_\epsilon(B_k)$ is the union of ϵ-balls $O_\epsilon(\mathbf{q}) = \{\ \mathbf{p} \mid \|\mathbf{p} - \mathbf{q}\| < \epsilon\}$ with the centers taken at all points $\mathbf{q} \in B_k$.

When the two surfaces A_k and B_k are tightly approximated by two toroidal patches T_A and T_B as follows

$$H(A_k, T_A) < \epsilon_A, \quad H(B_k, T_B) < \epsilon_B,$$

within two-sided Hausdorff distances $\epsilon_A, \epsilon_B > 0$, the condition $h(A_k, B_k) < \epsilon$ can be tested by checking

$$h(T_A, T_B) < \epsilon_T,$$

or equivalently

$$T_A \subset O_{\epsilon_T}(T_B),$$

where $\epsilon_T = \epsilon - \epsilon_A - \epsilon_B > 0$. Alternatively, we can also check if $A_k \subset O_{\epsilon_T + \epsilon_A}(T_B)$, to show $h(A_k, B_k) < \epsilon$. The offset volume $O_{\epsilon_T}(T_B)$ for a toroidal patch T_B is much simpler than $O_\epsilon(B_k)$ bounded by non-rational offset surfaces of B_k. The geometric simplicity of T_B is the main source of acceleration in our HD computation.

When we take $\epsilon > 0$ as a global lower bound $\underline{h} = d(\mathbf{p}, B)$ of $h(A, B)$, for some $\mathbf{p} \in A \setminus B$, we can conclude

$$h(A_k, B) \le h(A_k, B_k) < \epsilon = \underline{h} = d(\mathbf{p}, B) \le h(A, B),$$

and remove A_k from further consideration as it is useless for the HD computation.

4. HD Computation using BVH

The control flow of our algorithm is basically about the same as the previous HD algorithm of Kim et al. [16], which is briefly mentioned in Section 4.1. The main difference is in the way of handling the cross boundary case including the polar point case. Section 4.2 first presents a simple technique for estimating an upper bound $\overline{h}_k$ of $h(A_k, B)$ for internal nodes A_k. After that, we consider the main technical issue of this paper by introducing a torus-based method to make a good estimation of the local HD $h(A_k, B)$ for leaf nodes A_k. For this purpose, we need an efficient technique for projecting many points from A_k to B, for which osculating toroidal patches of B play a crucial role. Section 4.3 briefly explains how to improve the HD approximation result to a precise HD solution.

4.1. Basic control flow

By switching the roles of A and B if necessary, we may assume $H(A, B) = h(A, B)$ and discuss how to remove the majority of A except a small neighborhood of the maximum deviation point $\mathbf{p}^* \in A$ (from B) such that $d(\mathbf{p}^*, B) = h(A, B)$. We start with a covering of A by a union of subpatches $\{A_k\}$, where $A = \cup A_k$. (Using a BVH tree for A, we start with the surface A itself stored in the root node.) A global lower bound $\underline{h}$ ($\le h(A, B)$) can be initially taken as $\underline{h} = \max_k d(\mathbf{p}_k, B)$, for some sample points $\mathbf{p}_k \in A_k$, e.g., the corner points of A_k. We delete all subpatches A_k from the set $\{A_k\}$ if there is an upper bound $\overline{h}_k$ (for the local HD $h(A_k, B)$), which is smaller than the global lower bound $\underline{h}$, i.e., when we have $\overline{h}_k < \underline{h}$. The remaining subpatches A_k are inserted to a priority queue, according to the priority $\overline{h}_k$, a local upper bound estimated as discussed in Section 3.1. The global upper bound $\overline{h}$ is set to the value $\overline{h}_k$ of the highest priority, i.e., the local upper bound for the subpatch A_k stored in the top element of the priority queue.

The top element A_k is supposed to have the highest possibility to contain the maximum deviation point $\mathbf{p}^* \in A$ with $d(\mathbf{p}^*, B) = h(A, B)$. To narrow down the area containing the point $\mathbf{p}^*$, the exact location of which is unknown yet, the top element A_k is deleted from the priority queue and subdivided into smaller pieces, say A_{k_i}, $(i = 1, 2, \cdots, n_k)$. (These are the child nodes of A_k in the BVH tree.) For each of A_{k_i}, the global lower bound $\underline{h}$ is updated to $\max(\underline{h}, d(\mathbf{p}_{k_i}, B))$, when a new corner point $\mathbf{p}_{k_i} \in A_{k_i}$ has a larger

deviation from B than the previous sample points of A. Moreover, an upper bound $\overline{h}_{k_i}$ is also estimated for each A_{k_i}. Into the priority queue, we then insert only those subpatches A_{k_i} with the priority $\overline{h}_{k_i} (> \underline{h})$. (The BVH traversal will resume later from the nodes corresponding to these A_{k_i}'s; on the other hand, the BVH traversals terminate at the nodes corresponding to the other A_{k_i}'s that are thrown away at this stage due to their condition: $\overline{h}_{k_i} \leq \underline{h}$.)

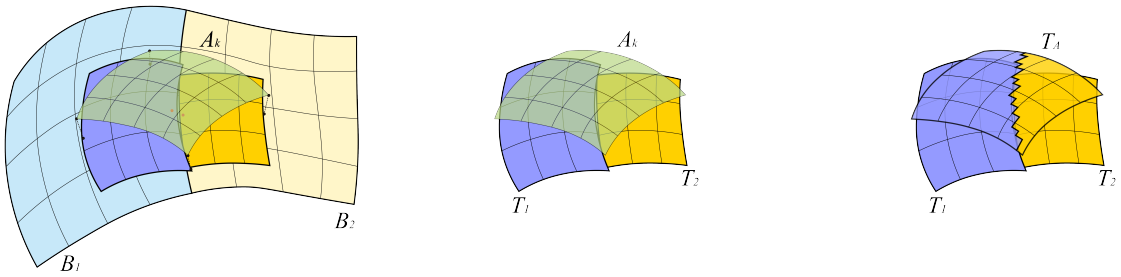
We repeat the same procedure until the difference $(\overline{h} - \underline{h})$ between the global upper and lower bounds is reduced within a given error tolerance, or the priority queue becomes empty.

4.2. Torus-based HD computation

In Section 3.1, we estimated an upper bound $\overline{h}_k$ for a surface A_k, assuming the four corner points are all projected to the same Bézier surface $B(s,t)$. (The point projection is computed using the BVH of $B(s,t)$.) When the four projections are made across the common boundary of two adjacent Bézier surfaces, as discussed in Section 3.2, it is difficult to formulate the difference surface $D(u,v)$ as a single Bézier surface. When the composite surface $B(s,t)$ is C^2-continuous across the common boundary, we can manage the estimation of an upper bound $\overline{h}_k$ using an error bounding technique of Filip et al. [9], which can be applied to C^2-continuous surfaces. The problem still remains across the common boundaries and around the polar points, where the surface $B(s,t)$ may not have C^2-continuity. In this case, we need to go down the BVH tree (of A) recursively all the way to the leaf nodes, where we can find the osculating toroidal patches that have projections to these cross-boundary areas. At the leaf level, trimmed osculating toroidal patches (dynamically generated on the Bézier surfaces $B_1(s,t)$ and $B_2(s,t)$) will replace the role of $B(s,t)$.

Before discussing the details of handling the leaf nodes, we need to estimate an upper bound $\overline{h}_k$ of $h(A_k, B)$ for each node A_k, so that a decision can be made on A_k: whether it should be inserted to the priority queue or removed from further consideration. It is not necessary to make an accurate estimation for internal nodes A_k. A fast estimation is often more important for a better performance of the algorithm. In the cross boundary case, we move some projection points to the other side of the boundary. For example, when only one corner of A_k is projected to $B_1(s,t)$ and the others are to $B_2(s,t)$, we move the projection point to $B_2(s,t)$. As discussed in Section 4.1, we further move one point to a new location so that their (s,t)-parameter points form a rectangle or a parallelogram. Then we can have a simple linear reparameterization of $s(u,v), t(u,v)$, and the matching surface $B_k(u,v) = B(s(u,v), t(u,v))$ will be of the same degree as $B(s,t)$. In an extreme case, we may take the surface $B_k(u,v)$ as a common boundary curve segment of two adjacent Bézier surfaces $B_1(s,t)$ and $B_2(s,t)$, or even as a polar point to which all control points of $B_k(u,v)$ collapse. Kang et al. [13; 14] also use similar techniques for a fast estimation of local upper bounds in some cases. (On the other hand, Kim et al. [16] skip this step for the cross-boundary case, by inserting A_k to the priority queue, which slows down the algorithm, in particular, for high-precision HD computations. Now, the difference surface $D(u,v) = A_k(u,v) - B_k(u,v) = \sum\sum \mathbf{d}_{ij} B_i(u) B_j(v)$ has the same degree as $A_k(u,v)$ and $B_k(u,v)$. As in Section 3.1, we take $\overline{h}_k = \max \|\mathbf{d}_{ij}\|$ as an upper bound $\overline{h}_k$ for $h(A_k, B)$. For a C^2-continuous composite surface $B(s,t)$, we estimate an upper bound $\overline{h}_k$ using the uniform sampling technique of Filip et al. [9], which often provides a tighter bound more efficiently.

At a leaf node A_k, with an osculating toroidal patch T_A, we consider the cross boundary case where a common boundary curve is shared by two adjacent Bézier surfaces B_1 and B_2 on a composite surface $B(s,t)$. The projections of four corners of A_k form a quadrangle Q in the st-parameter space of $B(s,t)$. We split Q into two pieces Q_1 and Q_2 in the domains of B_1 and B_2, respectively. Two osculating toroidal patches T_1 and T_2 are then computed at the centers $\mathbf{q}_1$ and $\mathbf{q}_2$ of Q_1 and Q_2, and trimmed along the parameter directions of B_1 and B_2. We can then proceed the HD computation by setting an upper bound $\overline{h}_k = \epsilon_A + \overline{h}_{T_A} + \epsilon_B$, where $\overline{h}_{T_A}$ is an upper bound for $h(T_A, T_1 \cup T_2)$. Figure 3 shows the construction of T_1 and T_2, and an estimation of $\overline{h}_{T_A}$ using a recursive subdivision of T_A. Note that the subdivision of T_A is more efficient than that of $A_k(u,v)$ and the projection of sample points to T_1 and T_2 is considerably more efficient than the projection to $B(s,t)$. This computational advantage is the main source of our acceleration over the previous HD algorithm of Kim et al. [16], where the Bézier suface patch $A_k(u,v)$ needs to be recursiveely subdivided across the common boundary of two adjacent Bézier surfaces $B_1(s,t)$ and $B_2(s,t)$.



(a) (b) (c)

Figure 3: Handling a leaf node A_k with the cross-boundary problem: (a) the four corners of A_k are projected to two adjacent Bézier surfaces B_1 and B_2, (b) two osculating toroidal patches T_1 and T_2 are constructed to replace the roles of B_1 and B_2, and (c) an upper bound $\overline{h}_{T_A}$ is computed by a recursive subdivision of T_A in a direction along the boundary of B_1 and B_2.

4.3. Precise HD computation

When the difference $(\overline{h} - \underline{h})$ between the global upper and lower bounds is reduced within a given error tolerance of the HD approximation, we move on to the final stage of computing a precise HD up to the machine precision, by solving the constraint equations of Elber and Grandine [8] for each subpatch A_k remaining in the priority queue until the queue becomes empty. In the numerical refinement step, the global lower bound $\underline{h}$ increases, which also accelerates the removal of some elements from the priority queue.

In the test examples of this paper, we switch to the precise HD computation with an initial solution within an error bound 10^{-9}. The goal is to reduce the HD computation error within $2.22 * 10^{-16}$, which is the machine precision for our implementation using 64 bit double precision floating-point numbers. We have observed that the constraint equations of Elber and Grandine [8] produce stable results until the numerical iterations reach the accuracy level of machine precision. After this, we have observed no consistent improvement of the computation results, as expected from the basics of numerical analysis.

5. Experimental Results

We have implemented the proposed HD computation algorithm in C++ on an Intel Core i7-10700K 3.8GHz CPU with a 128GB main memory and an NVIDIA Geforce RTX2080. To demonstrate the performance of our approach, we have tested the implemented algorithm for twelve freeform geometric models (of Figure 4), including (a)–(d) four general surfaces, (e)–(h) four surfaces of revolution, and (i)–(l) four surfaces of linear extrusion. For each model, the unit length is taken as the diagonal length of the minimum bounding box for the model. Table 1 shows some details of these freeform models and the statistics on the construction of their BVH trees. The model names are in the second row, and the number of Bézier surfaces for each model is given in the third row. The fourth row shows the maximum error bound ϵ_A (or ϵ_B) for the approximation of each leaf node of A (or B) by an osculating toroidal patch T_A (or T_B). Note that the summation $\epsilon_A + \epsilon_B = 2 \cdot 10^{-10}, 4 \cdot 10^{-10}$ of approximation errors from both surfaces A and B should be sufficiently smaller than the HD approximation error bound $\epsilon = 10^{-3}, \cdots, 10^{-9}$. This explains why we need to generate large BVH trees for general freeform surfaces, as shown in the 5–7th rows, by the statistics on the number of nodes, the memory size, and the BVH build time. Compared with the BVH trees for general surfaces, the surfaces of revolution and linear extrusion have many advantages both in space and time, as shown in the rest of this table.

Considering the main applications of HD in shape recognition and similarity tests, it would be ideal to test the HD computation for two almost identical (but unnoticeably different) models. However, it is difficult to generate two such models, in particular, when we want a pair of models for which the HD is known a priori. (The ground truth is important for checking the reliability of HD computation results.) Thus, taking a practical approach, we generate an almost identical model from the other by changing the position and orientation of the original model only slightly. In the special case of applying a pure

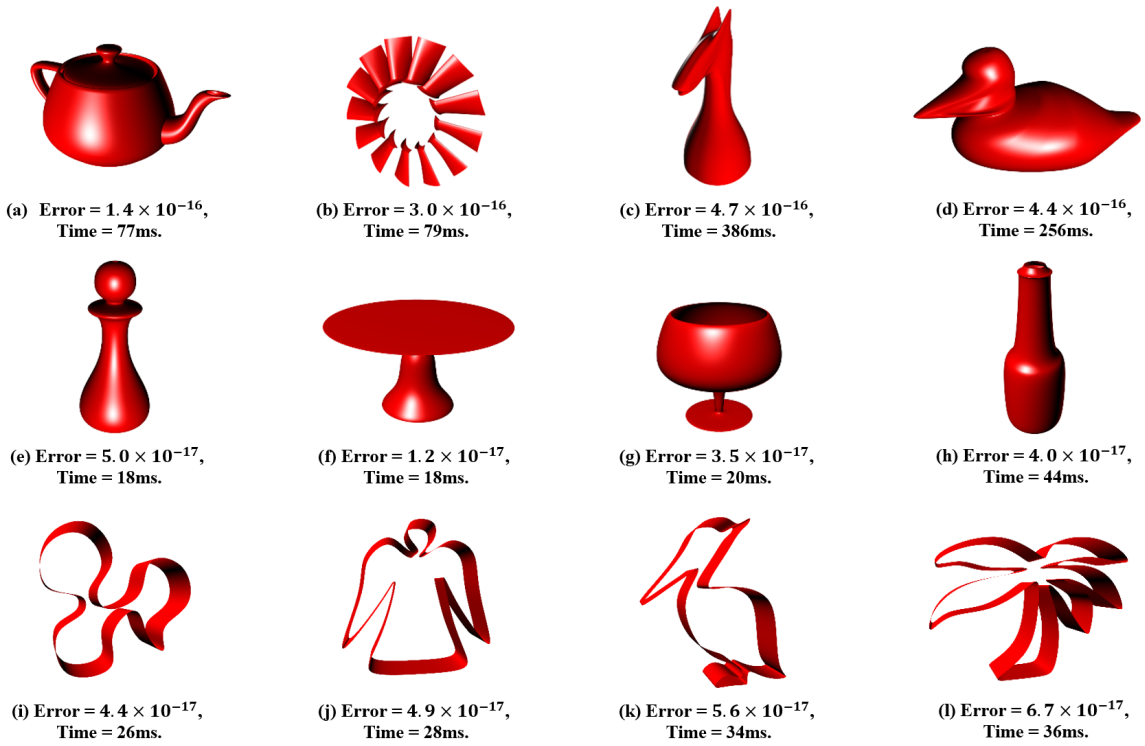


Figure 4: Twelve test models and their precise HD computation results.

	General				Revolution				Extrusion			
	Teapot	Turbine	Knight	Duck	Pawn	Table	Cup	Bottle	DuckExt	Angel	Bird	Tree
# Béziers	28	156	144	330	10	10	10	10	60	72	82	88
ϵ_A (or ϵ_B)	2e-10	2e-10	2e-10	2e-10	1e-10	1e-10	1e-10	1e-10	1e-10	1e-10	1e-10	1e-10
# Nodes	58,399,407	51,894,391	91,934,047	84,627,847	9,295	6,699	9,227	7,801	25,519	31,423	30,349	42,767
Size	56GB	49GB	88GB	81GB	8MB	6MB	8MB	7MB	22MB	27MB	26MB	37MB
Time	21min	10min	36min	22min	43msec	32msec	42msec	39msec	127msec	155msec	164msec	208msec

Table 1: Statistics on the BVH construction results. For general freeform models, parallel processing has been used.

translation to the model (with a fixed orientation), there is an important advantage. The ground truth of the exact HD is known a priori. (The distance of translation is exactly the same as the exact HD of the two models under consideration.) The provision of exact solutions is essential in measuring the precision of HD computation results, in particular, since our goal is to reduce the computation errors within machine precision. In Figure 4, each model is shown together with the HD approximation error and the computing time, which demonstrates that the precise HD computation of our algorithm produces solutions (within machine precision) for the special case of pure translations, for which the exact HDs are known a priori.

By changing the orientation of one model only slightly, we can also test the generic case, where the exact HD is unknown. As the HD computation is the most interesting for two almost identical models, we start with translations and rotations that introduce the maximum deviations in a similar distance range to the accuracy of the HD computation, and then repeat the same computations using tighter error bounds, and finally switch to precise HD computations with results within machine precision.

Figures 5–7 report the test results for twelve freeform geometric models. In each test, we consider a pair of identical models, and translate only one model by a distance $d > 0$ along 26 different directions $\mathbf{n}$, and approximate the HD within an error bound $\epsilon > 0$. The translation distance is fixed to $d = 0.001$, the

26 directions $\mathbf{n} = (n_x, n_y, n_z)$ are taken from all combinations of $n_x, n_y, n_z = -1, 0, 1$, except $\mathbf{n} = \mathbf{0}$, and the error bound $\epsilon > 0$ is iteratively reduced from 10^{-3} to 10^{-9} in seven steps: $\epsilon = 10^{-3}, \cdots, 10^{-9}$. We also consider the rotation of one model, followed by the 26 translations considered above. The rotation is by angle $\theta = 10^{-3} \cdot \pi$ about an axis $\mathbf{l}$, passing through the model center in the direction of $\mathbf{n}$. Finally, we consider 26 pure rotations about the different axes $\mathbf{l}$. In total, we have 728 ($= 26 * 28$) different configurations of (position, orientation) for one model with respect to the stationary copy of itself at the standard position and orientation. For each relative configuration, the same HD computation is repeated for a given error bound $\epsilon > 0$. A total of 728 HD computation results are averaged to produce the HD reported in this paper. The same experiments are then repeated for different values of $\epsilon = 10^{-3}, \cdots, 10^{-9}$, and all other freeform geometric models.

In Figures 5–7, each graph shows the HD approximation results for seven different levels of error bound $\epsilon = 10^{-3}, \cdots, 10^{-9}$. For the precise HD computation, the error in the level of machine precision can be measured only when the exact HD is known a priori. Thus we first consider the special test case where the two models have the same shape and orientation and the exact HD is given as the distance of translation. The HD error reported (below the rendered image of each model) is the average of 26 HD computation results for this special case. On the other hand, the HD time is the average computing time for 728 precise HD computations. The graph on the right shows in black lines the computing time taken in the HD approximation within an error bound $\epsilon > 0$. The red and blue lines show the values of $\overline{h}$ and $\underline{h}$, the global upper and lower bounds for the HD, respectively, which are computed as the result of HD approximation, so that they satisfy the termination condition: $\overline{h} - \underline{h} < \epsilon$

We have considered so far only the pairs of identical models with different positions or orientations. Using random perturbations of control points within a small distance, we can generate pairs of freeform models of slightly different shape. Figure 8 reports the test results, applied to two such pairs of models, where one Teapot is randomly perturbed within a distance 10^{-2} (and 10^{-3}). Because of the large BVH size, we increased the approximation error ϵ_A (or ϵ_B) for the Teapots to $2 \cdot 10^{-9}$, so that the BVHs for two Teapots under comparison can be loaded together to the main memory of 128GB. This explains why we have computed the HD approximation only up to the precision of 10^{-8} in this test. On the other hand, thanks to the reduced BVH sizes, we can compute the HD approximation about twice faster than the previous test, reported in Figure 5(a), for a pair of identical Teapots (sharing the same BVH). Nevertheless, this does not necessarily mean that the precise HD can also be computed more efficiently. In the case of perturbation 10^{-3}, we have observed that the precise HD took about twice more computing time, due to more elements left in the priority queue as the result of HD approximation with lower precision. To test the HD computation on a large set of pairs of non-identical freeform models, we have generated a deformable surface of revolution by interpolating a sequence of planar profile curves with a continuously changing planar curve. Applying a continuous rigid-body motion to the surface, we can generate a dynamically changing surface of revolution, in shape, position, and orientation, all at the same time. Figure 9 shows some snapshots for a pair of continuously changing surfaces of revolution thus generated. Some snapshots from their HD computation are shown in Figure 10. Moreover, Figure 11 reports the HD results in 1000 frames from the continuous deformation of two surfaces of revolution. The black line presents the HD as a continuous function of the frame numbers. Depending on the relative motion and deformation of the two models, the HD often reaches two or three times higher values than the model size. The red and blue lines are for the times taken in the HD computation and the BVH construction, respectively.

6. Conclusions

We have presented an efficient algorithm for computing the precise HD between two freeform geometric models, which is a significant improvement in precision over the previous HD algorithms. The main focus of this work is on computing the HD in machine precision, which has made the HD algorithm less ideal for the performance in computing speed. In future work, we plan to explore a new direction of the HD computation, in which we would like to provide a systematic way of balancing the precision and efficiency in HD computation, at least for some important applications in practice.

Acknowledgments

We would like to thank anonymous reviewers for their invaluable comments. This research was supported in part by the European Union Horizon 2020 research and innovation programme, under grant agreement No 862025, and in part by the National Research Foundation of Korea (No. NRF-2019R1A2C1003490, NRF-2019K1A3A1A78112596).

References

[1] Alt, H., Guibas, L., 1999. Discrete geometric shapes: Matching, interpolation, and approximation. *Handbook of Computational Geometry*, Elsevier Science.

[2] Alt, H., Scharf, L., 2008. Computing the Hausdorff distance between curved objects. *Int. J. Comput. Geometry Appl. 18*, 4, 307–320.

[3] Aspert, N., Santa-Cruz, D., Ebrahimi, T., 2002. MESH : Measuring errors between surfaces using the Hausdorff distance. *Proc. of the IEEE Int'l Conf. in Multimedia and Expo (ICME)*, Vol. 1, pp. 705–708, Lausanne, Switzerland, August 26-29.

[4] Bai, Y.-B., Yong, J.-H., Liu, C.-Y., Liu, X.-M., Meng, Y., 2011. Polyline approach for approximating the Hausdorff distance between two planar free-form curves, *Computer-Aided Design 43*, 6, 687–698.

[5] Barton, M., Hanniel, I., Elber, G., Kim, M.-S., 2010. Precise Hausdorff distance computation between polygonal meshes, *Computer Aided Geometric Design 27*, 8, 580–591.

[6] Chen, X.-D., Ma, W., Xu, G., Paul, J.-C., 2010. Computing the Hausdorff distance between two B-spline curves, *Computer-Aided Design 42*, 12, 1197–1206.

[7] Cignoni P., Rocchini C., Scopigno, R., 1998. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2, 167–174.

[8] Elber G., Grandine T., 2008. Hausdorff and minimal distances between parametric freeforms in R^2 and R^3. *Advances in Geometric Modeling and Processing*, F Chen and B Jüttler (Eds.), *Procs. of the 5th Int'l Conf.*, GMP 2008, Hangzhou, China, April 23-25, 2008. Lecture Notes in Computer Science 4975, Springer, pp. 191–204.

[9] Filip, D., Magedson, R., Markot, R., 1986. Surface approximations using bounds on derivatives. *Computer Aided Geometric Design 3*, 4, 295–311.

[10] Guthe, M., Borodin, P., Klein, R., 2005. Fast and accurate Hausdorff distance calculation between meshes. *Journal of WSCG*, V. Skala (ed.), vol. 13, pp. 41–48.

[11] Hanniel, I., Krishnamurthy, A., McMains, S., 2012. Computing the Hausdorff distance between NURBS surfaces using numerical iteration on the GPU. *Graphical Models 74*, 4 (2012), pp. 255–264, the special issue of *Geometric Modeling and Processing (GMP2012)*, June 18–22, 2012, Huangshan, China.

[12] Jüttler, B., 2000. Bounding the Hausdorff distance of implicitly defined and/or parametric curves. *Mathematical methods in CAGD*, Oslo 2000, pp. 1–10.

[13] Kang, Y., Kyung, M.-H., Yoon, S.-H., Kim, M.-S., Elber, G., 2018. Fast and robust Hausdorff distance computation from triangle mesh to quad mesh in near-zero cases, *Computer Aided Geometric Design 62*, 91–103.

[14] Kang, Y., Yoon, S.-H., Kyung, M.-H., Kim, M.-S., Elber, G., 2019. Fast and robust computation of the Hausdorff distance between triangle mesh and quad mesh for near-zero cases, *Computers & Graphics 81*, 61–72.

[15] Kim, Y.-J, Oh, Y.-T., Yoon, S.-H., Kim, M.-S., Elber, G., 2010. Precise Hausdorff distance computation for planar freeform curves using biarcs and depth buffer. *The Visual Computer 26*, 6–8, 1007–1016.

[16] Kim, Y.-J., Oh, Y.-T., Yoon, S.-H., Kim, M.-S., Elber, G., 2013. Efficient Hausdorff distance computation for freeform geometric models in close proximity. *Computer-Aided Design 45*, 2, 270–276.

[17] Krishnamurthy, A., McMains, S., and Hanniel, I., 2011. GPU-accelerated Hausdorff distance computation between dynamically deformable NURBS surfaces, *Computer-Aided Design 43*, 11, 1370–1379.

[18] Kurnosenko, A., 2013. Biarcs and bilens, *Computer Aided Geometric Design 30*, 3, 310–330.

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

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

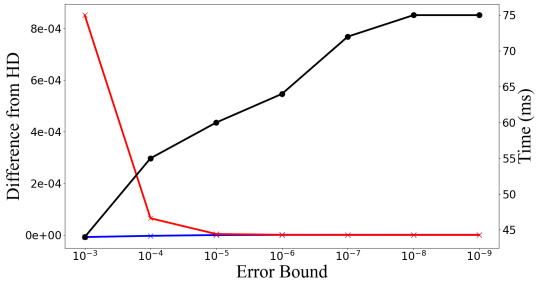
[21] Park, Y., Son, S., Kim, M.-S., Elber, G., 2020. Surface-surface-intersection computation using a bounding volume hierarchy with osculating toroidal patches in the leaf nodes, *Computer-Aided Design 45*, 2, 270–276.

[22] Rucklidge, W., 1996. *Efficient Visual Recognition using the Hausdorff Distance*, Lecture Notes in Computer Science, Vol. 1173, Springer-Verlag.

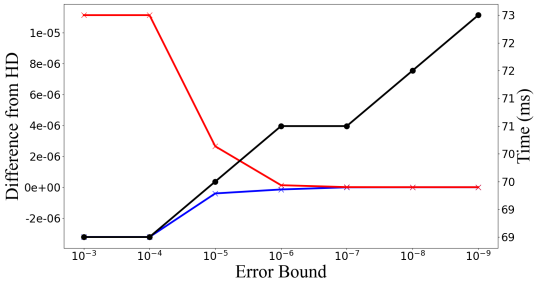
[23] Sir, Z., Feichtinger, R., Jüttler, B., 2006. Approximating curves and their offsets using biarcs and Pythagorean hodograph quintics, *Computer-Aided Design 38*, 6, 608–618.

[24] Son, S., Yoon, S.-H., Kim, M.-S., Elber, G., 2020. Efficient minimum distance computation for solids of revolution. *Computer Graphics Forum*, the special issue of Eurographics 2020.

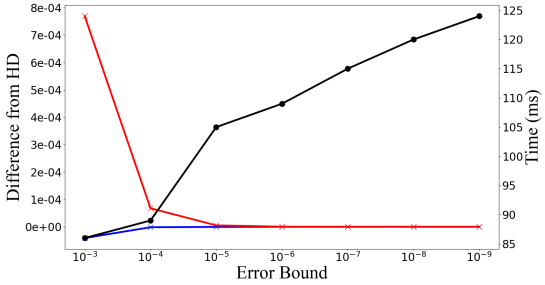
[25] Tang, M., Lee, M., Kim, Y.J., 2009. Interactive Hausdorff distance computation for general polygonal models, *ACM Trans. on Graphics 28*, 3, Article 74, (SIGGRAPH 2009).



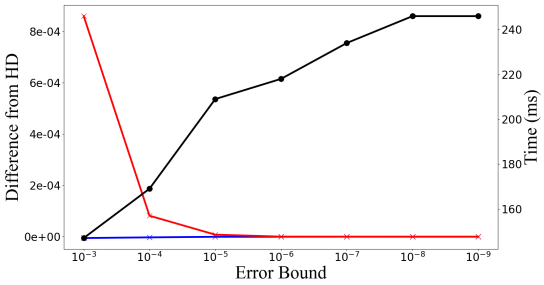
(a) Teapot



(b) Turbine

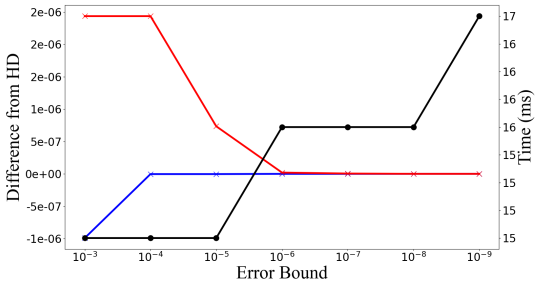


(c) Knight

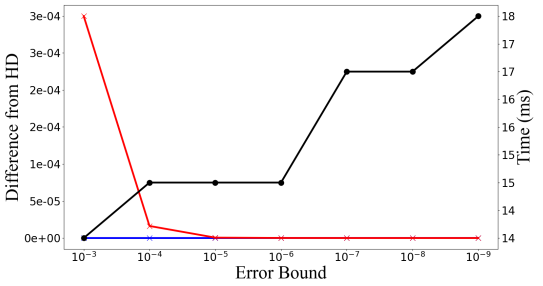


(d) Duck

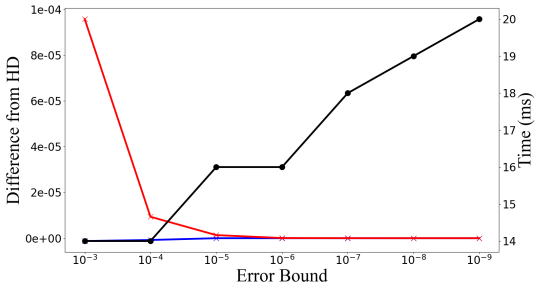
Figure 5: Experimental results for general freeform surfaces. The blue and red lines are the values for $\underline{h}$ and $\overline{h}$, the global lower and upper bounds, when their difference $\overline{h} - \underline{h}$ is reduced below the given error bound $\epsilon > 0$. The black line reports the HD computation time.



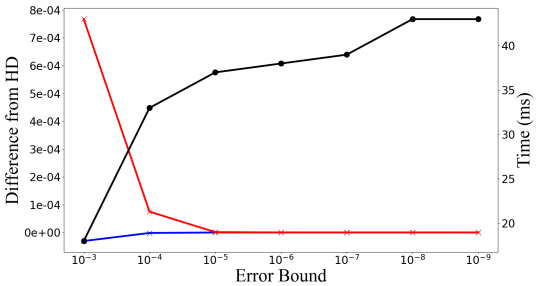
(a) Pawn



(b) Table

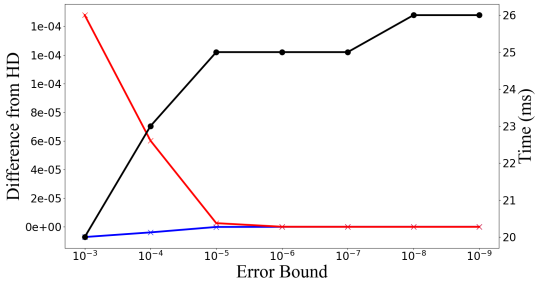


(c) Cup

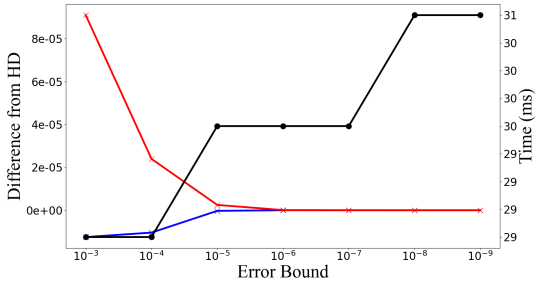


(d) Bottle

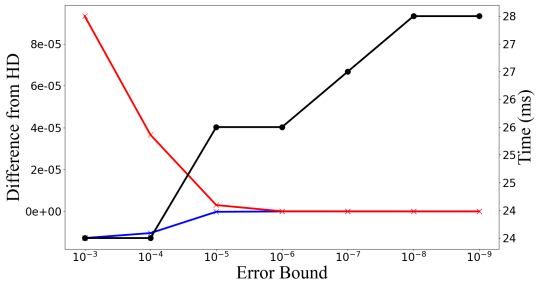
Figure 6: Experimental results for surfaces of revolution.



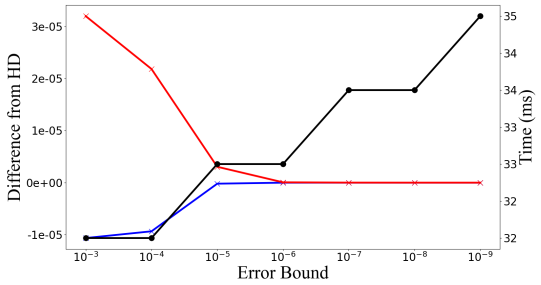
(a) DuckExt



(b) Bird

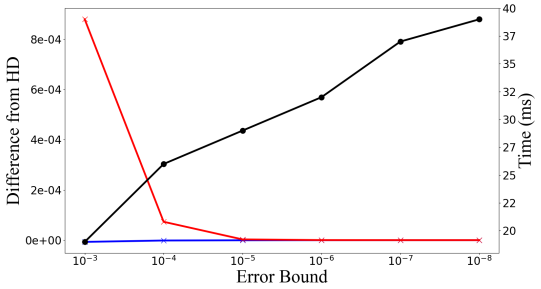


(c) Angel

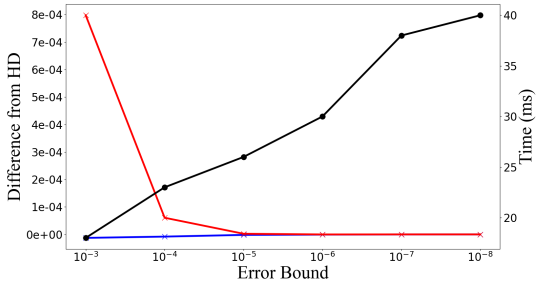


(d) Tree

Figure 7: Experimental results for surfaces of linear extrusion.



(a) Teapot with perturbation up to 10^{-2}



(b) Teapot with perturbation up to 10^{-3}

Figure 8: The HD computation between the Teapot model and a perturbed one. Random perturbations of up to: (a) 10^{-2} and (b) 10^{-3}, are applied to the control points of one Teapot.

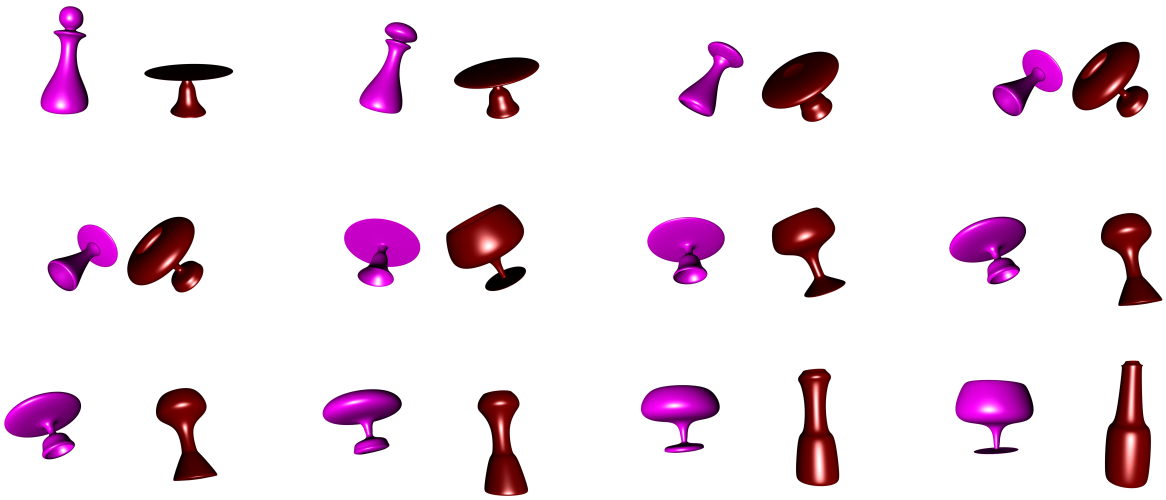
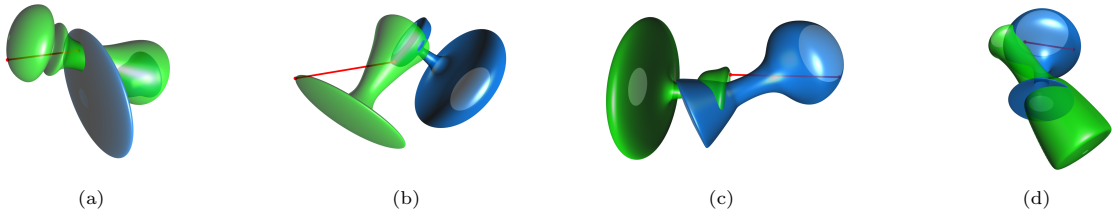


Figure 9: Snapshots from a pair of deformable surfaces of revolution.



(a) (b) (c) (d)

Figure 10: Snapshots from the HD computation for a pair of continuously changing surfaces of revolution.

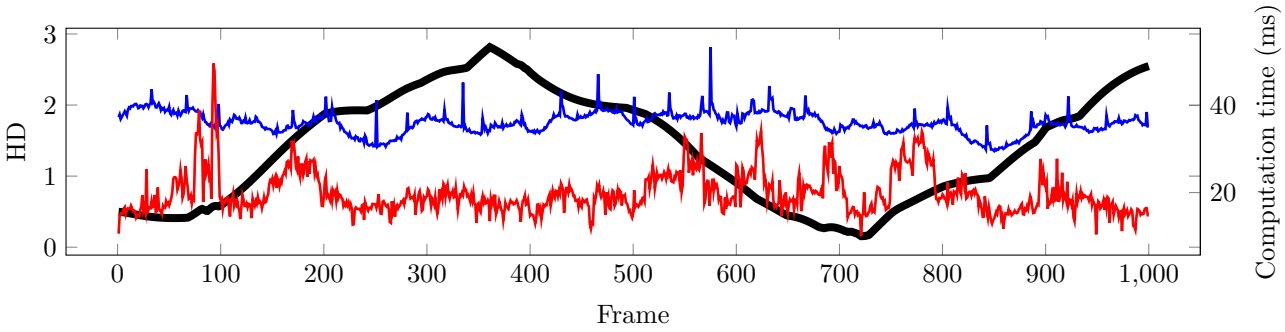


Figure 11: HD computation for a pair of continuously changing surfaces of revolution, within an error bound $\epsilon = 10^{-8}$. The HD computing time is shown in red, and the BVH construction time is in blue. The computed HD is shown in black.