M.S. THESIS

Efficient Geometric Algorithms Using Osculating Toroidal Patches

최대 접촉 토러스 패치를 이용한 효율적인 기하학적 알고리즘

FEBRUARY 2021

서울대학교 대학원

전기 · 컴퓨터공학부

손 상 현

M.S. THESIS

Efficient Geometric Algorithms Using Osculating Toroidal Patches

최대 접촉 토러스 패치를 이용한 효율적인 기하학적 알고리즘

FEBRUARY 2021

서울대학교 대학원

전기 · 컴퓨터공학부

손 상 현

Efficient Geometric Algorithms Using Osculating Toroidal Patches

최대 접촉 토러스 패치를 이용한 효율적인 기하학적 알고리즘

지도교수 김 명 수

이 논문을 공학석사 학위논문으로 제출함

2020 년 11 월

서울대학교 대학원

전기 · 컴퓨터공학부

손 상 현

손 상 현의 공학석사 학위논문을 인준함

2020 년 12 월

위 원 장 _____ 신 영 길 _____ (인)

부위원장 _____ 김 명 수 _____ (인)

위 원 _____ 이 영 기 _____ (인)

Abstract

We present efficient geometric algorithms that are based upon toroidal patches. To begin with, we present to use osculating toroidal patches to approximate a regular surface and propose a reparameterization method for the approximating toroidal patches. Then, we show that the toroidal patches can approximate special kinds of freeform parametric surfaces that are built upon planar profile curves much more effectively than general surfaces.

Thanks to these precise toroidal patches, we can construct a very compact bounding volume hierarchy for a parametric surface. With the bounding volume hierarchy, we can perform fast and precise point projection, i.e., minimum distance computation from a point to the surface. Also, we can easily find binormal lines, i.e. lines that connect two geometric entities orthogonally, between toroidal patches and use them to find meaningful distance measures for parametric surfaces. We show that we can find such binormal lines easily by finding binormal lines between circles in space.

Using these fundamental toroidal geometric operations, we present an efficient minimum distance computation algorithm for solids of revolution. This algorithm accelerates the minimum distance computation 10-100 times faster than conventional method. Also, we propose an efficient Hausdorff Distance computation algorithm that is applicable to various kinds of parametric surfaces. We can find the Hausdorff Distance, almost up to machine precision, without much cost increase. Even though these algorithms follow conventional frameworks in large, they exhibit much better precision and efficiency than

previous methods because of the toroidal patches that we use in our hierarchy.

Keywords: Toroidal patch, Bounding Volume Hiearchy, Point projection, Binormal computation, Minimum distance computation, Hausdorff Distance computation

Student Number: 2019-27608

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In geometric modeling, freeform parametric surfaces are widely used, e.g., Bezier surfaces and B-spline surfaces [14]. Compared to other forms of geometry, they have many useful properties. For instance, they have continuity property, unlike discrete representations such as triangular mesh. Therefore, we can analyze them with techniques from classical differential geometry [11]. Also, users can control the shapes of surfaces without much difficulty by handling control points of the surfaces. This is rather difficult for other kinds of polynomial parametric surface, such as power-basis surface [41].

In general, we can represent arbitrary shapes very effectively with the freeform parametric surfaces. For instance, Utah teapot in Figure 1.1(a), which is frequently used in computer graphics research, can be represented with several pieces of B-spline surfaces. In this work, we call them general freeform

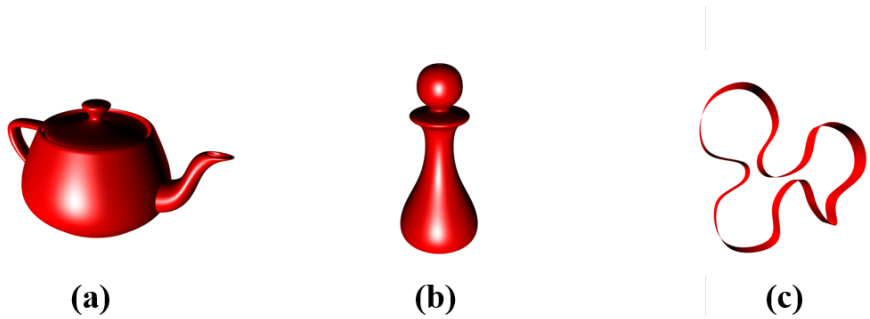


Figure 1.1 Examples of freeform parametric surfaces

parametric surfaces to distinguish them from special types of freeform parametric surfaces. Those special types include surface of revolution and surface of linear extrusion.

First, we can define a surface of revolution by rotating a curve around an arbitrary axis. A pawn that is used in chess game(Figure 1.1(b)) is typical example. Then, the control points and knot vectors have to follow special structure to represent the rotation, and we have to use non-uniform rational B-spline basis functions to define it [14]. The surface of revolution is widely used in many fields of physics and engineering, especially for the design of many industrial parts in computer-aided design. For instance, general milling tool used in CNC machining is a surface of revolution [5].

Additionally, we can define a surface of linear extrusion by linearly extruding a curve along certain direction in space. Then, the control points are located along the direction, starting from the original control points of the curve. Thanks to the simplicity of it, surface of linear extrusion is also widely used in the field of computer-aided design. For instance, a surface of linear

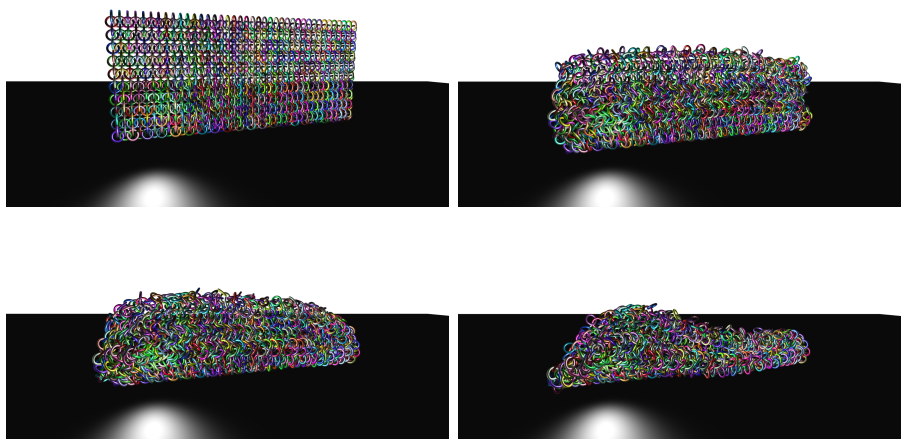


Figure 1.2 Falling simulation of interlocked rings

extrusion can be used as a template to create multiple identical objects. See the surface of linear extrusion in Figure 1.1(c) which has the shape of duck.

These surfaces are first designed with Computer-Aided Design(CAD) tools. Then, we can perform various operations with these surfaces. For instance, we can simulate their movements or responses to outer stimuli with various analysis techniques. If we assume they are rigid bodies, we can simulate their movements with dynamics simulators, such as Open Dynamics Engine [47] (Figure 1.2). On the other hand, if they can change their shapes, we can simulate the deformations with isogeometric analysis [10]. Also, we can store the models in the database and compare each other to retrieve specific models that we want. All these operations require efficient distance computation algorithms. To be specific, in simulations, minimum distance, or collision detection algorithm is bottleneck in general. To compare 3D models, we need an efficient algorithm to compute Hausdorff Distance between the target models [1], and this task requires a lot of computation cost, too. Therefore, we must enhance

the efficiency of these operations to improve the quality of the applications above.

1.2 Research Objectives and Contributions

In this work, we aim to accelerate the minimum distance computation for solids of revolution. Also, we aim to enhance both the precision and the efficiency of Hausdorff Distance computation for all the freeform parametric surfaces introduced in Section 1.1.

To achieve this goal, we will use Bounding Volume Hierarchy (BVH) that are based on toroidal patches. BVH is a data structure that is used to accelerate various geometric operations. BVH maintains a tree, of which nodes have bounding volumes that bound certain subset of the target object [20]. We will use toroidal patches in the leaf nodes to approximate the surface. Then using the upper bound of the approximation error as the radius of a sphere, we will generate toroidal patch swept spheres that totally bound the given surface.

With this special BVH, we can perform fundamental geometric operations that play vital role in the aformentioned algorithms quite efficiently.

To be specific, the minimum distance computation for solids of revolution requires efficient binormal computation for toroidal patches. Binormal lines are lines that pass through two geometries orthogonally [13]. We will show that we can reduce the problem to finding binormal lines for circles in space, which is much easier problem.

The Hausdorff Distance computation for freeform surfaces requires rapid and precise point projection operation. Since toroidal patches approximate surfaces very effectively, we can project points to the surfaces very precisely

with the toroidal patches. Also, since toroidal patches admit simple point projection algorithm, we can find foot points on the surfaces very efficiently [36].

With these observations, we will show that BVH based on toroidal patches can be used to achieve our goals in the end. Therefore, our contribution can be summarized as follows:

- We introduce a BVH based on toroidal patches. For a general regular surface, we will suggest a reparameterization method that relates the parameters of the original surface and the parameters of the approximating toroidal patches. For surface of revolution and surface of linear extrusion, we propose toroidal approximation that is based upon biarc approximation of a planar profile curve. We will show that this special BVH can approximate given surface much more effectively than conventional BVH, which is based upon convex bounding volumes.

- We propose fundamental toroidal operations that we can apply to our BVH. First, based on the point projection algorithm of Liu et al. [36], we propose a point projection algorithm that can be applied to our BVH. Then, we suggest an efficient binormal computation algorithm for toroidal patches. The efficiency comes from the fact that the problem can be reduced to much easier problem, which is equivalent to solving 8^{th} degree polynomial.

- Based on the basic toroidal operations, we show that we can greatly accelerate the minimum distance computation for solids of revolution. Also, we present a Hausdorff Distance computation algorithm that is much more efficient than previous algorithms. We prove these accomplishments with various test cases.

1.3 Thesis Organization

In this chapter, we have addressed our research background and objectives. Chapter 2 will provide some basics of freeform parametric surfaces and torus. Then, in Chapter 3, we will review previous works on BVH, minimum distance computation, and Hausdorff distance computation. In Chapter 4, we introduce BVH based on toroidal patches. We will discuss toroidal patch approximation methods for different kinds of parametric surfaces, and then provide basic geometric operations than can be done with toroidal patches. In Chapter 5, we introduce the minimum distance computation algorithm for solids of revolution, and the Hausdorff Distance computation algorithm that is applicable to all the different kinds of freeform parametric surfaces. We will prove our algorithm's effectiveness with various test cases. In the end, in Chapter 6, we conclude this thesis.

Chapter 2

Preliminaries

2.1 Freeform Parametric Surface

When representing a parametric surface in geometric modeling, each of the coordinates of a point on a surface is formulated as an explicit function of two independent parameters [41] :

$$S(u, v) = (x(u, v), y(u, v), z(u, v))$$

$$(u_0 \leq u \leq u_1, v_0 \leq v \leq v_1)$$

Usually, polynomial functions are widely used for the coordinate functions. It stems from the fact they are easy to handle with computer, and also easy to understand.

2.1.1 Bézier Surface

Among many polynomial parametric surfaces, Bézier surface is widely used in geometric modeling as it conveys geometric insight about the shape of the surface and thus provide natural means to design interactively [41].

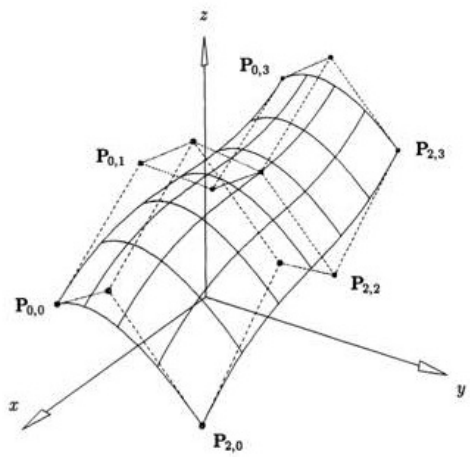


Figure 2.1 Bézier surface with its control points $P_{i,j}$ [41]

A Bézier surface of degree m in the u direction and degree n in the v direction is defined by Bernstein polynomials $B_{i,m}(u), B_{j,n}(v)$ and control points $\mathbf{P}_{i,j}$ [15] :

$$S(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(u)B_{j,n}(v)\mathbf{P}_{i,j}(0 \leq u,v \leq 1)$$

$$B_{i,m}(u) = \frac{m!}{i!(m-i)!}u^i(1-u)^{m-i}$$

$$B_{j,n}(v) = \frac{n!}{j!(n-j)!}v^j(1-v)^{n-j}$$

This thesis will mainly focus on Bézier surface and B-spline surface, which is comprised of multiple Bezier surfaces [14], to deal with a general freeform parametric surface of arbitrary shape.

2.1.2 Surface of Revolution

A surface of revolution is generated by rotating a profile curve, or genetrix, around an axis of rotation.

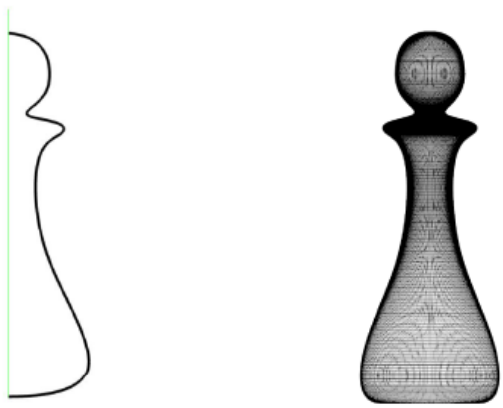


Figure 2.2 Surface of revolution and its profile curve

Figure 2.2 shows a pawn-shaped surface of revolution and its profile curve. It is generated by rotating a profile curve in XY-plane around the Y axis. All the surfaces of revolution that we use in this work are defined in this way. We use u-parameter to determine the planar point on the profile curve, and v-parameter to determine the degree of rotation. Therefore, a surface of revolution can be defined as follows :

$$S(u,v) = (x(u)cos(v), y(u), z(u)sinv)(u_0 \leq u \leq u_1, 0 \leq v \leq 2\pi)$$

$$Genetrix : C(u) = (x(u), y(u))(u_0 \leq u \leq u_1)$$

In this work, we will also discuss a solid of revolution, which is defined by rotating a planar region around an axis of rotation. We simply extend a surface of revolution to a solid of revolution by assuming the planar region is bounded by the axis of rotation (which is Y axis in this work), parallel lines to X axis, and the original profile curve.

2.1.3 Surface of Linear Extrusion

A surface of linear extrusion is generated by extruding a profile curve, or genetrix, along certain direction linearly.

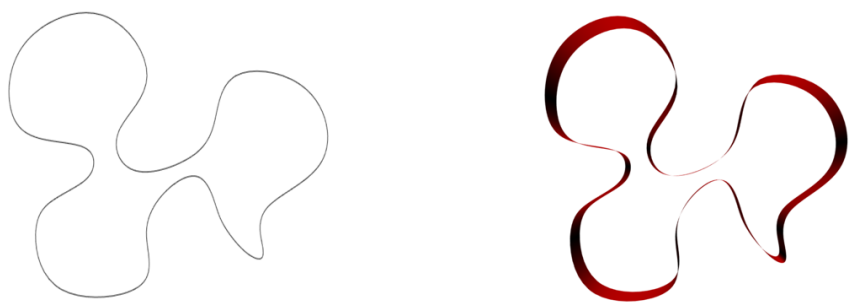


Figure 2.3 Surface of linear extrusion and its profile curve

In this work, we use a surface of linear extrusion defined by extruding a profile curve in XY-plane along Z axis. Figure 2.3 shows an examples of such surface of linear extrusion. As a surface of revolution, we use u-parameter to determine the planar point on the profile curve and v-parameter to determine the degree of extrusion. Then, a surface of linear extrusion is defined as :

$$S(u,v) = (x(u), y(u), v)(u_0 \leq u \leq u_1, v_0 \leq v \leq v_1)$$

$$Genetrix : C(u) = (x(u), y(u))(u_0 \leq u \leq u_1)$$

2.2 Torus

Torus is a surface generated by rotating a planar circle of radius r about a straight line belonging to the plane of the circle and at a distance $R > r$ away from the center of the circle [11]. We call R as the major radius of the torus, and r as the minor radius of the torus.

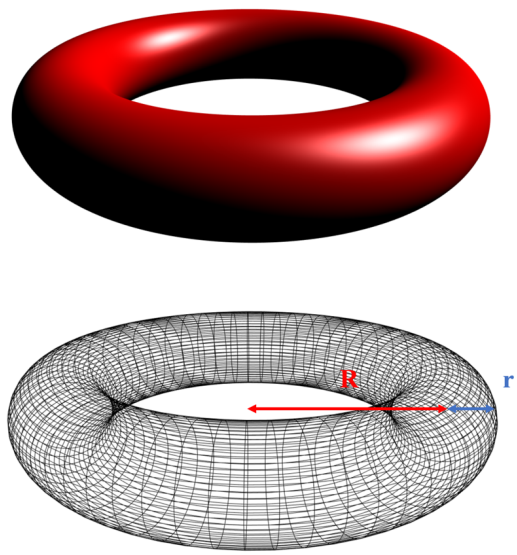


Figure 2.4 Torus of major radius R and minor radius r

When we consider a torus that is generated by rotating a planar circle in XY plane about Y axis, the torus is defined as follows :

$$T(u, v) = ((R + rcosu)cosv, (R + rcosu)sinv, rsinu)(0 < u, v < 2\pi)$$

Torus has following useful geometric properties [11] :

- A torus is regular surface. Therefore, it has continuous partial derivatives

of all orders in its domain.

- A torus is well segmented into regions with positive, negative, and zero Gaussian curvatures (Figure 2.5).
- Principal directions and principal curvatures of an arbitrary point on a torus is obtained immediately.

Thanks to these properties, torus can be used as a geometric primitive to use in accelerating geometric algorithms for parametric surfaces. This thesis will mainly deal with a toroidal patch, which is a subset of a torus. In Figure 2.6, a toroidal patch that is extracted from a full torus, which has major radius of 1 and minor radius of 0.25, by restricting its domain to $-0.1\pi \leq u \leq 0.1\pi, -0.1\pi \leq v \leq 0.1\pi$, is illustrated in red.

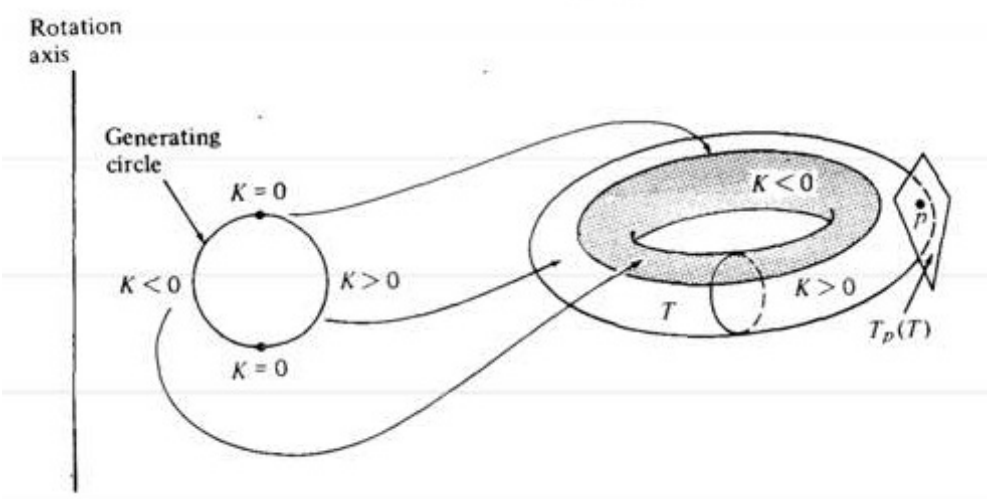


Figure 2.5 Segmentation of torus by Gaussian curvature K [11]

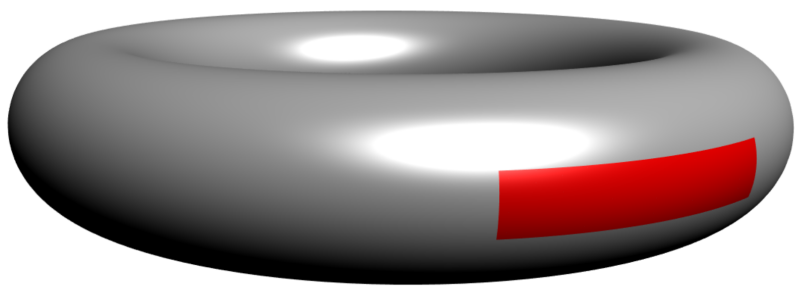


Figure 2.6 Example toroidal patch

Chapter 3

Related Work

3.1 Bounding Volume Hierarchy

Bounding Volume Hierarchy(BVH) is a data structure that is used to accelerate various geometric algorithms. BVH maintains a tree, which is comprised of nodes that have bounding volumes to bound certain subset of the target object. Generally, a BVH should meet four basic requirements to be used in geometric algorithms [6, 20]:

- the hierarchy conservatively approximates the object, each level representing a tighter fit than its parent;
- for any node in the hierarchy, its children should cover the parts of the object covered by the parent node;
- the hierarchy should be created in a predictable automatic manner, not requiring user interaction;

- the bounding volumes within the hierarchy should fit the original model as tightly as possible, representing it to a high degree of accuracy.

Many geometric primitives have been proposed for constructing the BVH that satisfy conditions above, and many of them were convex bounding volumes. They include Spheres [6, 20, 43], Axis Aligned Bounding Boxes(AABB) [4], Oriented Bounding Boxes(OBB) [18], Discrete Oriented Polytopes(k-DOPs) [28], and Sphere Swept Volumes(SSV) [32]. Since convex geometric primitives are easy to handle and there are efficient minimum distance computation algorithms for them [17, 35], they were frequently used even when the given object was in non-convex shape [43].

However, we cannot deny that such bounding volumes lack approximation power when it comes to non-convex parts of a shape. To overcome this issue, non-convex bounding volumes were proposed. For planar curves, non-convex bounding regions such as fat arcs [45] and bounding circular arcs(BCA) [38] were proposed. They were used to accelerate curve-curve intersection [45], offset trimming [25, 33], and voronoi diagram construction [34] for planar curves.

For surfaces in space, a spherical shell was suggested as the first non-convex bounding volume [30]. However, since it can only imitate one of the principal curvatures at the approximation point, it cannot approximate hyperbolic surfaces that have saddle points. Liu et al. [36] overcame this issue with osculating tori, as torus can approximate every principal curvature and direction at the approximation point. In this work, we use the concept of osculating torus, prove its additional mathematical properties, and use it to construct our BVH.

3.2 Minimum Distance Computation

There are many efficient algorithms to compute minimum distance between two convex geometric primitives [17, 35], and also their extensions [7]. Based on these basic algorithms, Larsen et al. [32] proposed a general paradigm for computing minimum distance between models represented in the BVH of sphere-swept bounding volumes. The paradigm is described in Section 5.1.1, and we follow this paradigam in our work, too.

There have been many variants of the algorithm, but the general frameworks were all same [8, 22, 23]. Among them, it is worth to note that Kim et al. [26] used Coons patches to approximate a surface accurately, and accelerated collision detection and minimum distance computation with the BVH. This shows that accurate approximation of surface and compact BVH plays important role in accelerating minimum distance computations. Based on this result, we show that our BVH based on toroidal patches can greatly accelerate minimum distance computation for solids of revolution.

3.3 Hausdorff Distance Computation

There have been many Hausdorff Distance computation algorithms for various forms of geometry, including polygonal meshes [2, 3, 9], parametric curves [21, 44], and parametric surfaces [13]. However, it has been very difficult to compute the Hausdorff Distance in real-time, especially when the two objects are in close proximity. In that case, due to large number of overlapping subsets from both objects, another method to compensate the computation overload was required, such as GPU [19, 29].

However, Kim et al. [27] and Kang et al. [24] made significant contribu-

tions to the Hausdorff Distance computation, by accelerating it to interactive speed. Especially, Kang et al. [24] showed that their algorithm can compute the Hausdorff Distance between polygonal mesh models precisely without much cost increase. These two approaches are both based on estimating the lower bound and upper bound of the Hausdorff Distance by point projection and parameter matching. In this work, we follow this approach, but use BVH based on toroidal patches to overcome issues that were not addressed in [27]. With this method, we show that we can compute the Hausdorff Distance between freeform parametric surfaces effectively, without significant increase of cost as [24].

Chapter 4

Bounding Volume Hierarchy

This chapter introduces a bounding volume hierarchy(BVH) that is based upon toroidal patches that approximate given freeform parametric surfaces.

4.1 Construction

Given a rectangular domain $R \in \mathbb{R}^2$ and a surface $S(u, v) : R \to \mathbb{R}^3$ defined on the domain, we can construct a BVH that bounds the surface with a precision of ϵ with the following recursive algorithm.

Algorithm 1: Recursive BVH Construction Algorithm

Result: BVH of surface S

input: S = Surface,

R = Domain of S to approximate,

ϵ = precision

1 Approximate $S(R)$ with a simple geometric primitive (Optional);

2 Approximate $S(R)$ with a toroidal patch T;

3 **if** *Error bound of* $T > \epsilon$ **then**

4 $[R_1, ...R_n]$ = Subdivisions of R;

5 Call this routine recursively for each subdivision $[R_1, ...R_n]$;

6 **else**

7 Set leaf flag and insert this node to tree;

8 **end**

In line 1, we approximate the specified region of the surface with a simpler geometric primitive than toroidal patch. For instance, we can use Rectangular Swept Sphere(RSS) [40] for a general freeform parametric surface, or truncated cylinder or cone for a surface of revolution [48]. Even they lack approximation power, they usually offer easier geometric operations than toroidal patches. Therefore, we can use them to cull out implausible candidate nodes at the beginning of many geometric algorithms.

In line 2, we approximate the specified region of the surface with a toroidal patch as described in the following sections, and compute the upper bound of approximation error. If the error bound is larger than the desired precision, we can further subdivide current domain and repeat the procedure until the desired precision is met.

Table 8 shows the BVH construction results for four different models that

	Teapot		Turbine		Knight		Duck	
ϵ	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
10^{-3}	11,935	1,935	29,523	1,923	21,211	3,367	19,667	3,299
	(7MB)	(2MB)	(17MB)	(2MB)	(12MB)	(3MB)	(11MB)	(3MB)
10^{-4}	142,871	8,991	150,515	8,267	179,227	14,119	225,179	13,603
	(82MB)	(9MB)	(87MB)	(8MB)	(103MB)	(13MB)	(130MB)	(13MB)
10^{-5}	1,333,335	42,143	2,135,171	39,987	1,806,283	64,847	1,835,531	61,755
	(768MB)	(40MB)	(1,229MB)	(38MB)	(1,040MB)	(62MB)	(1,057MB)	(59MB)

Table 4.1 BVH construction results for four different models : (a) BVH based on quadrilateral, (b) BVH based on toroidal patch

we use for testing our Hausdorff Distance computation algorithm in Section 5.2 (Figure 5.5). We constructed BVHs for three different levels of precision, $10^{-3}, 10^{-4}$, and 10^{-5} of unit length. The unit length equals to the diagonal length of the bounding box of each model. Each number in the cell denotes the number of nodes in the hierarchy, with its data size.

Note that the BVHs constructed with toroidal patches have much less number of nodes and thus take much less space than the BVH constructed with quadrilaterals [16]. Also note that BVHs based on toroidal patches become much more compact than BVHs based on quadrilaterals when it comes to high precision. In the following section, we explain how to approximate a regular surface with a toroidal patch.

4.2 Toroidal Patch Approximation

4.2.1 Regular surface

Let us consider a situation where we have to approximate a subset of a regular surface $S(u, v)(u_1 \leq u \leq u_2, v_1 \leq v \leq v_2)$. At the mid-parameter $u_0 = \frac{u_1+u_2}{2}, v_0 = \frac{v_1+v_2}{2}$, we compute following values :

- Surface point $\mathbf{Q} = S(u_0, v_0)$
- Normal vector $\mathbf{n}$ at $\mathbf{Q}$
- Two principal curvatures κ_0, κ_1 and corresponding unit principal direction vectors $\mathbf{e_0}, \mathbf{e_1}$ at $\mathbf{Q}$

If we exclude exceptional cases where the point $\mathbf{Q}$ is a parabolic or planar point, we can get a unique torus T that approximates the given surface at $\mathbf{Q}$ using above values :

1. If $\|\kappa_0\| > \|\kappa_1\|$, exchange both κ_0, κ_1 and $\mathbf{e_0}, \mathbf{e_1}$
2. If $\kappa_0 > 0$, multiply $\mathbf{n}$, κ_0, and κ_1 with -1
3. Create a torus T with center at $\mathbf{Q} + \frac{\mathbf{n}}{\kappa_0}$, the axis direction $\mathbf{e_1}$, the major radius $\frac{-1}{\kappa_0} + \frac{1}{\kappa_1}$, and minor radius $\frac{1}{\|\kappa_1\|}$
4. Determine the torus parameter (s_0, t_0) that satisfies $T(s_0, t_0) = \mathbf{Q}$.
   (a) If $\mathbf{Q}$ is an elliptic point, $(s_0, t_0) = (0, 0)$.
   (b) If $\mathbf{Q}$ is an hyperbolic point, $(s_0, t_0) = (0, \pi)$.

Then the torus $T(s, t)$ has second order contact to the surface $S(u, v)$. This means that the torus has identical principal directions, principal curvatures, and parallel normal vector with the surface S at $\mathbf{Q}$ [36].

When we create the osculating torus T of the surface S in this way, we can use a reparameterization method that relates the parameters of S and the parameters of T in the Appendix A. The reparameterzation method is based on the first and second derivatives of the surface S and the osculating torus T at the contact point $\mathbf{Q}$.

In Figures 4.1, 4.2, 4.3, we present some illustrations which show the effectiveness of toroidal patch approximation compared to the quadrilateral approximation of [16]. In the left column of the illustrations, target surface is rendered in wireframe and approximating primitives are rendered in blue. In the right column, surface points and their corresponding points on approximating primitives are connected with red line.

In Figures 4.1, 4.2, we can see that toroidal patches approximate target surface much more precisely than quadrilaterals for both elliptic and hyperbolic contact points. Also, in Figure 4.3, even when the target surface is in triangular shape, toroidal patch can imitate its topology while quadrilateral fails. It is because the reparameterization method of Theorem 2 uses up to second order derivatives at the contact point.

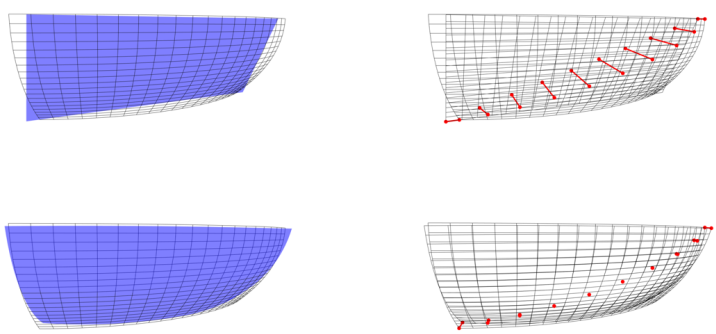


Figure 4.1 Comparison of approximation using quadrilateral(Up) and toroidal patch(Down) : Elliptic contact point

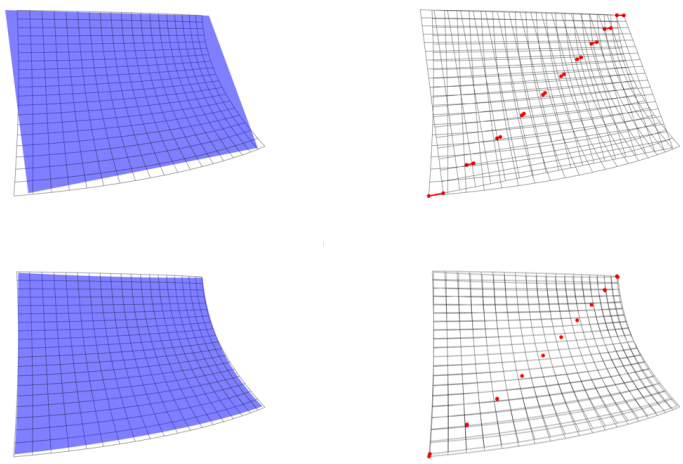


Figure 4.2 Comparison of approximation using quadrilateral(Up) and toroidal patch(Down) : Hyperbolic contact point

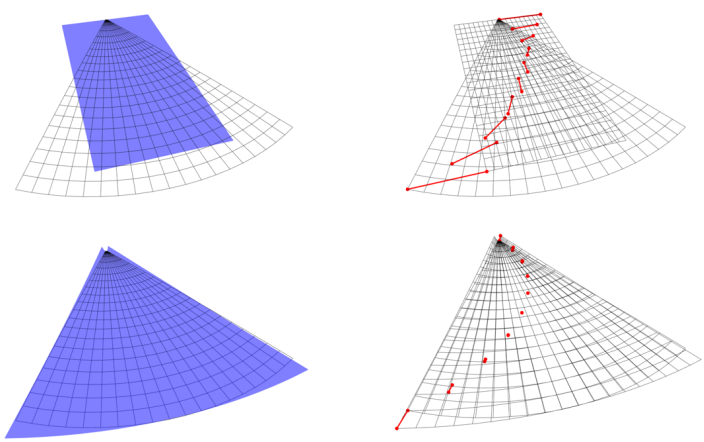


Figure 4.3 Comparison of approximation using quadrilateral(Up) and toroidal patch(Down) : Triangular region

4.2.2 Surface of Revolution

If we use a regular curve as a profile curve, we can apply Theorem 2 to the surface of revolution, too. However, we can approximate a surface of revolution with toroidal patches in much more efficient manner.

As a surface of revolution is defined by a planar profile curve, we can segment the curve into G^1-continuous circular arcs within a desired precision $\epsilon > 0$[46]. By rotating the circular arcs around Y axis, we gain a G^1-continuous toroidal patches that approximate the surface of revolution within the same precision [48].

Figure 4.4 Bilens approximation for a planar curve

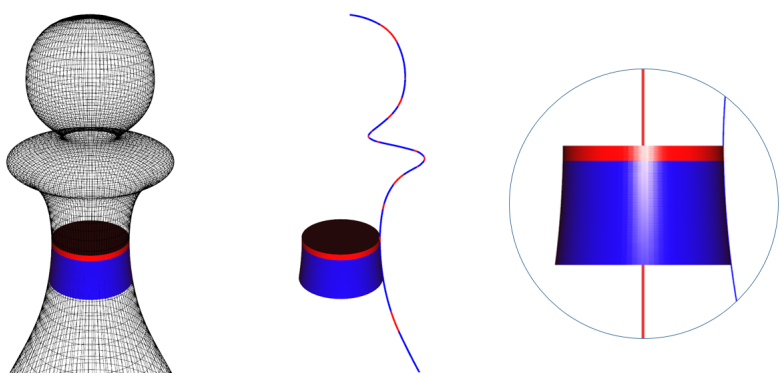


Figure 4.5 Toroidal patch approximation for a pawn model

In this work, we first segmenet the profile curve into pieces of spiral curves. Spiral curves are curves that are free of X, Y, and curvature extremums, and also free of zero curvature points. After that, we approximate each spiral curve with bilens, which tigtly bounds the spiral curve by interpolating both the two endpoints and the two tangent directions of it [31]. Figure 4.4 shows the result of bilens approximation for a planar curve, and Figure 4.5 shows some toroidal patches that approximate a pawn model. Note that the toroidal patches are generated by rotating red and blue circular arcs around the Y axis.

Also note that the toroidal patches generated in this way have $[0, 2\pi]$ as its v-parameter domain naturally and we do not have to care for approximation along v-parameter direction. Therefore, the number of toroidal patches needed to approximate a surface of revolution is much less than that needed to approximate a general regular surface.

4.2.3 Surface of Linear Extrusion

As a surface of revolution, we can approximate a surface of linear extrusion much more efficiently with cylindrical patches(Figure 4.6), which are extensions of toroidal patches.

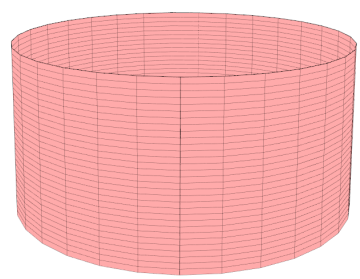


Figure 4.6 Cylindrical patch

The approximation procedure for this case is similar to that of a surface of revolution. We approximate the planar profile curve with circular arcs with a desired precision $\epsilon > 0$. Then, we just extrude the circular arcs along Z axis and generate cylindrical patches. Then the G^1-continuous cylindrical patches approximate the surface of linear extrusion within the same precision ϵ.

Note that we do not have to care for approximation along v-direction as the case of surface of revolution. Therefore, the number of cylindrical patches needed to approximate a surface of linear extrusion is also very small compared to that needed to approximate a general regular surface.

4.3 Toroidal Operations

Toroidal patches admit efficient fundamental geometric operations. In this work, we present two of them : Point projection and binormal computation.

4.3.1 Point Projection

We can perform point projection on toroidal patches very efficiently. When it comes to a general toroidal patch T, not approximating any surface, we can perform point projection from an arbitrary point $P \in \mathbb{R}^3$ in two steps [36].

1. Project P on the major circular arc and get parameter s
2. Project P on the minor circular arc determined by the parameter s and get parameter t

Liu et al. [36] showed that we can accelerate point projection on a surface with osculating toroidal patches that approximate the surface. They first project a test point P on to the osculating toroidal patches, which are made

on the fly, and have constant pre-defined (s, t) domain. Then they retrieve the original surface's parameters by solving overconstrained quadratic equation obtained from the second order Taylor's expansion.

Our point projection algorithm also uses toroidal patches that approximate the target surface. However, our algorithm differs from that of Liu et al. [36] in two facets.

- We make toroidal patches that approximate the target surface with pre-specified precision before running point projection algorithm. Also, those toroidal patches' (s, t) domains are determined by the reparameterization relationship of Theorem 2. Therefore, each toroidal patch's domain differs from one another.

- When performing parameter inversion to retrieve target surface parameter (u, v), we use the reparameterization relationship of Theorem 2.

  After we obtain the torus parameter (s_0, t_0), we can find the corresponding surface parameter (u_0, v_0) by solving following set of equations :

$$s_0 = c_1u^2 + c_2v^2 + c_3uv + c_4u + c_5v + c_6$$

$$t_0 = d_1u^2 + d_2v^2 + d_3uv + d_4u + d_5v + d_6$$

  We can solve the equations using Newton-Rhapson iteration method [42]. This solution differs from that of Liu et al. [36] as it solves two equations to get two unknowns (u, v).

  Note that sometimes there is no corresponding surface parameter (u_0, v_0) for the torus parameter (s_0, t_0) in the surface's domain R due to the misalignment of parameter spaces between the surface and the toroidal

patch. In this case, we can find the surface parameter (u_0, v_0) by minimizing the following energy function :

$$\min_{(u,v)\in R} \|T_2(u,v) - P\|$$

$$(T_2(u,v) = T(s(u,v), t(u,v))$$

4.3.2 Binormal Computation

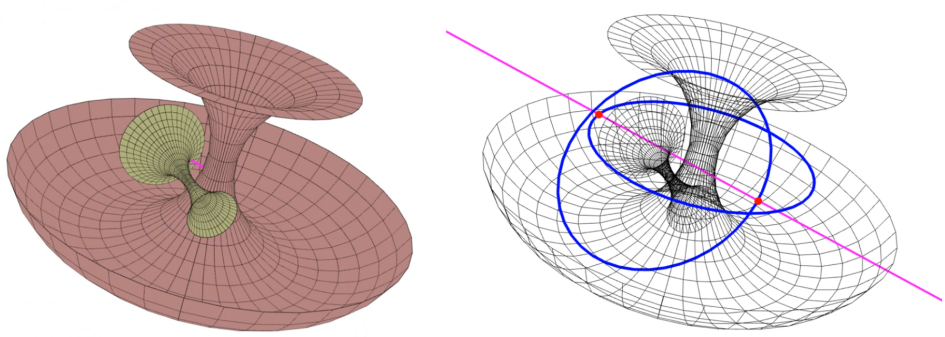


Figure 4.7 A binormal line of two toroidal patches

Binormal line is a line that passes through two geometric entities orthogonally [13]. When computing minimum, maximum, or Hausdorff distance between geometrical models, this concept of binormal line becomes crucial as such distance measures mainly occur on the binormal lines.

Given surfaces $F(u,v), G(s,t) \in \mathbb{R}^3$, we can find binormal lines of them by

solving following set of equations :

$$\langle F_u(u,v), F(u,v) - G(s,t)\rangle = 0$$
$$\langle F_v(u,v), F(u,v) - G(s,t)\rangle = 0$$
$$\langle G_s(u,v), F(u,v) - G(s,t)\rangle = 0$$
$$\langle G_t(u,v), F(u,v) - G(s,t)\rangle = 0$$

This method also applies to a pair of tori. However, when it comes to tori, we can find the binormal lines of them much more efficiently by finding binormal lines of their major circles. It stems from the fact that the binormal lines of the two tori must lie on the binormal lines of their major circles [48]. In Figure 4.7, we can see that the pink binormal line of two toroidal patches lie on a binormal line of their major circles rendered in blue.

Given a torus $T(u,v) \in \mathbb{R}^3$ with major radius of R and minor radius of r, we can formulate the major circle of the torus as follows :

$$C(v) = (Rcosv, Rsinv, 0)$$

Therefore, when there are two tori $T_1(u,v), T_2(s,t) \in \mathbb{R}^3$, we can find binormal lines of them with binormal lines of their major circles $C_1(v), C_2(t)$. It means that we can find them by solving following set of equations :

$$\langle C_{1,v}(v), C_1(v) - C_2(t)\rangle = 0$$
$$\langle C_{2,t}(t), C_1(v) - C_2(t)\rangle = 0$$

Since the number of unknowns and equations decrease by half, we can find binormal lines between tori much more efficiently than general surfaces. Furthermore, it is known that there are at most eight binormal lines exist between circles $C_1, C_2 \in \mathbb{R}^3$. In Figure 4.8, all of those binormal lines are rendered in

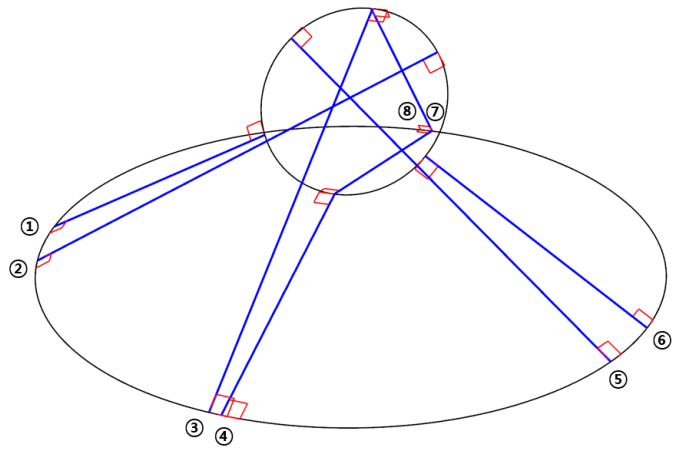


Figure 4.8 Eight binormal lines between two circles in space

blue. Additionally, we can find all of them by solving a 8^{th}-degree polynomial equation [12, 39, 48]. Usually, we can solve the polynomial in several microseconds, which shows the efficiency of this solution. Using this efficient binormal computation algorithm, we could run physics simulation that involves hundreds of interlocked rings in real-time (Figure 1.2).

Even more, when it comes to two toroidal patches that have restricted u, v domain, we can find their binormal lines more efficiently. Since the domains of possible solutions are restricted, we can cull out those that do not possess possible solutions while solving the 8^{th}-degree polynomial. In this work, we adopted the polynomial solver of Machchhar et al. [37] and accelerated the process.

Chapter 5

Geometric Algorithms

In this section, we show how we can accelerate specific geometric algorithms with BVH based on toroidal patches. First, we show minimum distance computation algorithm for solids of revolution can be accelerated a lot with the BVH. Then, we present Hausdorff distance computation algorithm for every kind of freeform parametric surfaces introduced in Section 2.1. The Hausdorff distance computation algorithm exhibits much faster speed and much more precise results than previous methods.

5.1 Minimum distance computation for solids of revolution

5.1.1 General Framework

BVH-based algorithms for minimum distance computation use priority queue to maintain potential pairs of nodes that can give the minimum distance be-

tween the models [32]. The priority of each pair is determined by the minimum distance between the bounding volumes stored in the nodes. The algorithm keeps track of the upper bound $\overline{h}$ and the lower bound $\underline{h}$ of the global minimum distance h and each time it computes the minimum distance between a pair of nodes, it updates the bounds. When the difference between $\overline{h}$ and $\underline{h}$ becomes less than the desired precision, we can terminate the algorithm and return $\frac{\overline{h}+\underline{h}}{2}$ as the minimum distance.

Generally, when we compute the minimum distance between a pair of nodes, we can get the upper bound and the lower bound of the minimum distance between the pair. Therefore, if the computed lower bound of minimum distance between a pair of nodes is larger than the upper bound of the global minimum distance, we can cull out the pair of nodes. Thanks to this property, we can cull out most pairs of nodes even when they are not leaf nodes.

5.1.2 Algorithm

A solid of revolution, which is an extended version of a surface of revolution, can be also approximated with toroidal patches like a surface of revolution. However, as it is a solid, we assume the approximating toroidal patches have solid inner parts. Also, in Algorithm 1, we assume truncated cylinder patches and conical patches also have solid inner parts. Therefore, they can be considered as truncated solid cylinders and cones.

When there are two BVHs that bound two different solids of revolution, we can find minimum distance between them with following algorithm :

Algorithm 2: BVH algorithm for minimum 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 Both nodes are leaf then
        Find binormal lines {b_i} between toroidal patches;
        if ∃{b_i} then
            Find minimum distance d from {b_i};
            ub = d + ε_A + ε_B;
            if ub < UpperBound then
                Upper Bound = ub;
            end
        end
    else
        d = Minimum distance between truncated cone/cylinder;
        lb = d − ε_A − ε_B;
        if lb < UpperBound then
            Insert every pair of children into Q with lb;
        end
    end
end
```

The Algorithm 2 differs from conventional BVH-based minimum distance

computation algorithm [32] in following points :

- We use truncated solids of cone or cylinder when one or both nodes in the pair are not leaf nodes. It is because the distance computation for toroidal patches, which includes binormal computation, is more expensive than for the simpler primitives. Even they lack approximation power, we can cull out unnecessary pairs of nodes in the top level very efficiently with the simple primitives.

- Our approximation methods guarantee two-sided Hausdorff distance bounding. It means that we can get the lower bound and upper bound of the minimum distance between the pair of nodes by subtracting or adding approximation errors (ϵ_A, ϵ_B) to the computed minimum distance. Therefore, in the Algorithm 2, we only consider the upper bound ub in the final step and just return the upper bound as the final minimum distance when the priority queue is empty.

- As we use biarc approximation for the profile curve, the circular arcs, and thus toroidal patches, are connected G^1-continuously. Thanks to this property, we do not have to consider edge cases where the minimum distance occurs on the edges of the toroidal patches. Therefore, in the Algorithm 2, if there are no binormal lines between toroidal patches, we can just ignore the pair of nodes.

5.1.3 Experimental Results

We have implemented our algorithm in C++ on an Intel Core i-7 6700 3.4GHz PC with a 16 GB main memory. We have tested the algorithm on five static solids of revolution($M_1 - M_5$, Figure 5.1) and one deformable solid of revolution

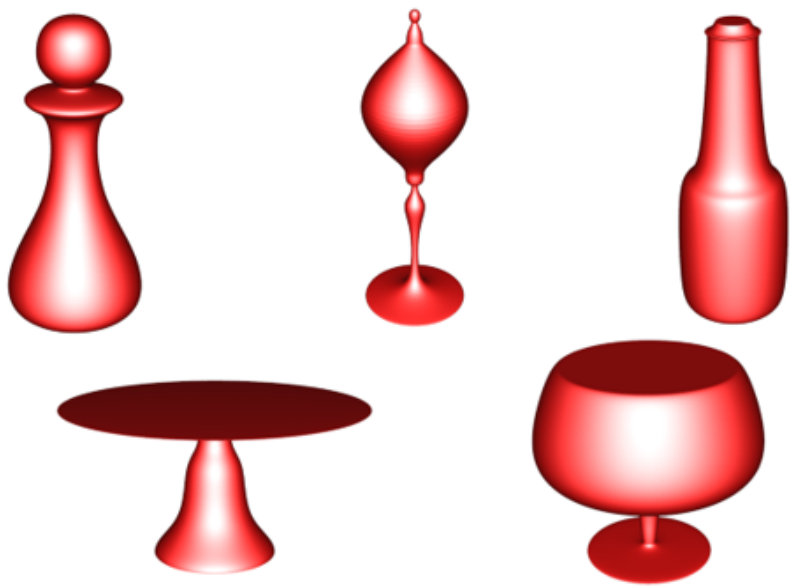


Figure 5.1 Static solid of revolution models

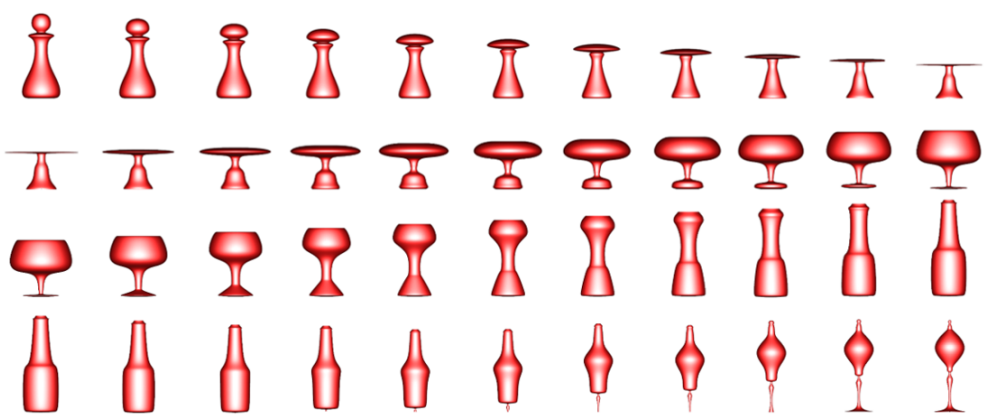


Figure 5.2 Deformable solid of revolution model

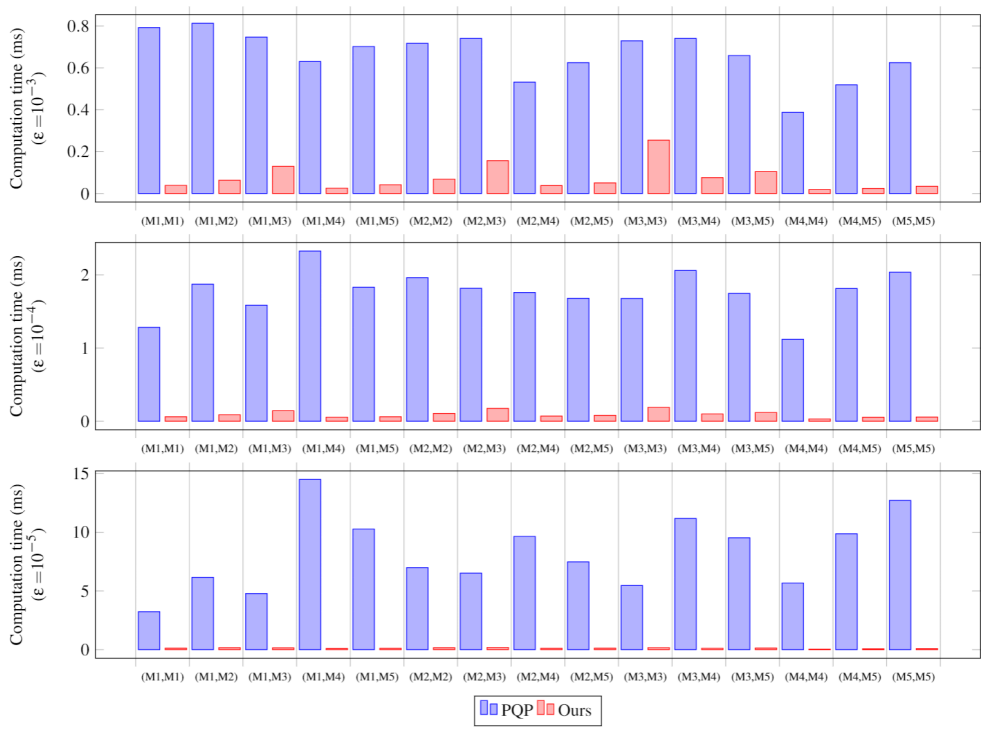


Figure 5.3 Comparison of the relative performance of our algorithm against the approach of Larsen et al. [32]

($M_{t_j}(j = 1, \cdots, 1000)$, Figure 5.2). We have generated the deformable solid of revolution by interpolating the profile curves of the five static models and selecting 1000 frames among them. In the following experiments, we have made quantative comparisons of our approach against conventional methods, represented by the algorithm of Larsen et al. [32], using the PQP library (http://gamma.cs.unc.edu/SSV/).

For each pair of static solid models $(M_i, M_j)(i \leq j)$, we have computed the minimum distances between the two models at 50,000 different frames taken from their continuous motion. Figure 5.3 reports a total of 15 test results thus

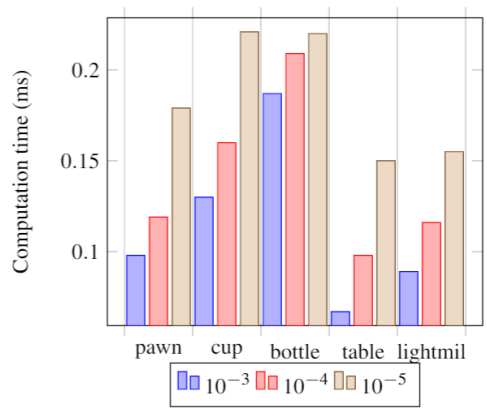


Figure 5.4 Average computation time (in ms) for the minimum distance between static models and a deformable model

computed. In the tests, three different values of precision($\epsilon = 10^{-3}, 10^{-4}, 10^{-5}$) were used. In Figure 5.3, the computation times of the conventional approach are represented in blue vertical bars, and the computation times of our approach are represented in red vertical bars. We also tested our algorithm by computing minimum distances between the five static models and the deformable model. Figure 5.4 shows the average computation time for the given pair of models (M_i, M_{t_j}).

The results show that our algorithm runs much faster than the conventional approach, often $10-100$ times faster, for static models. Also, the results prove that our algorithm can handle deformable solids of revolution in real-time, which had not been possible with previous methods. This huge increase of efficiency comes from both the compact data structure and the efficient binormal computation algorithm for toroidal patches.

5.2 Hausdorff distance computation

5.2.1 General Framework

Given two surfaces A and B, the one-sided Hausdorff Distance(HD) from A to B is defined as :

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

The (two-sided) Hausdorff Distance is then defined as :

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

In this work, we will follow the overall scheme that was proposed by Kim et al. [27] to compute the Hausdorff Distance. Generally speaking, the one-sided Hausdorff Distance $h(A,B)$ satisfies :

$$h(\underline{A},\overline{B}) \leq h(A,B) \leq h(\overline{A},\underline{B}),$$

for any $\underline{A} \subset A \subset \overline{A}$ and $\underline{B} \subset B \subset \overline{B}$, and we can use this property to compute the Hausdorff Distance.

To get the lower bound $\underline{h} = h(\underline{A},\overline{B})$ of the Hausdorff Distance, we construct the subset $\underline{A}$ using sample points $\mathbf{p}_i$ of A and the superset $\overline{B}$ using the BVH of B. That is, by projecting the sample points $\mathbf{p}_i$ to the BVH of B, we can compute the lower bound of the desired Hausdorff Distance. As described in Section 4.3.1, we can perform this point projection procedure very precisely and efficiently with BVH based on toroidal patches.

To get the upper bound $\overline{h} = h(\overline{A},\underline{B})$, we construct the superset $\overline{A}$ using the BVH of A and the subset $\underline{B}$ using the matching surface patches B_i to the given surface patch A_i. To be specific, as we use a sample point $\mathbf{p}_i$ of each

surface patch A_i when computing the lower bound $\underline{h}$, we can find a surface patch B_i in B that contains the closest point in B to $\mathbf{p}_i$. Then, we assume that the surface patch B_i is the closest surface patch in B to A_i and use it as the subset $\underline{B}$.

After we find the matching surface patch, we find the upper bound $\overline{h}$ by sampling. To be specific, we first project four corners of current surface patch A_i to the matching surface patch B_i. Then we sample sample points from A_i and find matching points in B_i with the information from the corner projection results. We compute the distance between all the matching point pairs, and use the maximum distance as the upper bound of $h(A_i, B_i)$.

We maintain a priority queue Q of which item contains a node in BVH of A and B. The priority of each item is determined by the upper bound of the one-sided Hausdorff Distance from the node to the other surface, which we call local Hausdorff Distance of the node. Therefore, we initialize the queue by inserting every leaf node in the BVH of A and B. Then we pop a item to update the lower bound and the upper bound of the Hausdorff Distance. If the computed upper bound of the local Hausdorff Distance is still larger than current lower bound of the global Hausdorff Distance, we have to push child nodes of current item into the queue. In this scheme, the upper bound of the global Hausdorff Distance is the upper bound of the local Hausdorff Distance of the first item in the queue. When the difference between the lower bound and the upper bound of the global Hausdorff Distance becomes less than the desired precision ϵ, we can stop the iteration and return the result.

5.2.2 Algorithm

Our Hausdorff Distance computation algorithm is comprised of two parts. The first part is a subroutine to process a specific node of single BVH. It computes the lower bound and the upper bound of the local Hausdorff Distance from the node to the other surface.

Algorithm 3: Algorithm to compute single node's HD

Result: Lower Bound and Upper Bound of HD for given node

input: Node N in BVH A or B

```
1  Sample point m = midpoint of N;
2  Lower Bound = Minimum distance from m to other surface's BVH;
3  Upper Bound = ∞;
4  Project corners of N to the closest surface patch found from Lower
     Bound computation and extract matching surface patch S;
5  for Every sample point p_i ∈ N do
6  |   Find matching point q_i ∈ S;
7  |   if ‖p_i − q_i‖ > UpperBound then
8  |   |   Upper Bound = ‖p_i − q_i‖;
9  |   end
10 end
```

We have to explain some details about the Algorithm 3.

- When the given node N is not leaf node, we can find matching point of a sample point $\mathbf{p}_i$ by bilinear reparameterization method [27]. This matching scheme is very efficient and provides safe upper bound of the local Hausdorff Distance. Also, it works quite well when the topology of both surface patch of N and the matching surface patch S is quadrilateral, which is often the case.

- If the given node N is leaf node, it usually means that the node falls in exceptional cases. In those exceptional cases, we cannot use aforementioned bilinear reparameterization method, as it hardly estimates precise matching point.

  The exceptional cases occur when the surface patch N spans various surface patches of the other surface, or the topology of the surface patches are not quadrilateral (often triangular). In these cases, we approximate the matching surface patch S with a toroidal patch, and then project sample points $\mathbf{p}_i$ onto the toroidal patch. Since foot points on the toroidal patch are much more precise matching points than those from bilinear reparameterization scheme, and the toroidal patch can approximate a surface patch even when it has triangular topology(Figure 4.3), we can deal with these cases effectively.

Now, we can use Algorithm 3 as a subroutine of Algorithm 4 to compute the Hausdorff Distance. Note that the lower bound and the upper bound of the global Hausdorff Distance is updated by the bounds of local Hausdorff Distance of each node. Also, in the final step where the difference between the lower bound and the upper bound falls below the desired precision, we can use algebraic method [13] to find precise Hausdorff Distance. Since the remaining items provide good candidates for precise Hausdorff Distance computation, we can use them as starting points for local numerical search.

Algorithm 4: BVH algorithm for Hausdorff Distance computation

Result: Hausdorff Distance

input : BVH A, B

Precision ϵ

output: HD

```
1  Lower Bound = −∞;
2  Upper Bound = ∞;
3  Insert every leaf node of A and B into priority Queue Q;
4  while Q is not empty do
5      Pop (item) from Q;
6      Upper Bound = item.upper_bound;
7      if Upper Bound < Lower Bound + ϵ then
8          Find precise HD from the remaining items in the Q;
9          return HD;
10     else
11         Create new items I_i from child nodes of current item;
12         Process every I_i with Algorithm 3;
13         if I_i.lower_bound > Lower Bound then
14             Lower Bound = I_i.lower_bound;
15         end
16         if I_i.upper_bound > Lower Bound then
17             Push I_i into the Q;
18         end
19     end
20 end
```

5.2.3 Experimental Results

We have implemented our Hausdorff Distance 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 algorithm for 12 different models of non-trivial complexity(Figure 5.5, 5.6, 5.7).

Tables 5.1, 5.2, 5.3 report the performance of our algorithm in computing the Hausdorff Distance between two identical models. In the first row, the name of the model and its number of Bézier patches are shown. In the second row, the data size of the BVH and its construction time is shown.

For a pair of models, we first tested the algorithm by translating one of the models along 26 different directions by length of 0.001. Then, we also rotated a model by radian angle of 0.001π and then translated it along the X axis by length of 0.001 to test the algorithm. The first test results are shown under the column (a) in the tables, and the second test results are shown under the column (b) in the tables. In the third row, the maximum size of the priority queue for each test case is also shown.

For each test case, we took various values from 10^{-3} to 10^{-9} of the model size as the tolerance of terminating the algorithm, i.e., we terminate when the difference $\overline{h} - \underline{h}$ is less than the specified value. From the 5^{th} to 11^{th} row, the tolerance, $\underline{h}$, $\overline{h}$, elapsed time (in ms), number of iterations, and the final size of the priority queue is reported for each test case. Then, in the last row, we reported the precise Hausdorff Distance computation result.

We can access our algorithm's correctness and efficiency with the experimental results :

- Correctness : In the case of pure translation, the exact Hausdorff Distance is the same as the distance of translation. Therefore, the Hausdorff Distance of test case (a) must be 0.001. We can see that our algorithm successfully bounds the value in given tolerance. Also, the precise Hausdorff Distance computation results show that our algorithm estimates the Hausdorff Distance very precisely, almost up to machine precision for every test case (a). Therefore, we can guarantee that our algorithm finds the true Hausdorff Distance.

- Efficiency : The results show that our algorithm finds the Hausdorff Distance for every model in interactive time, mostly in real-time. Also, we can see that there are not so significant increase of computation cost when we decrease the tolerance. This means that our algorithm does not fall into exceptional cases that hinder the algorithm. This efficiency comes from not only the BVH based on toroidal patches that permit rapid point projections, but also torus approximations in leaf nodes that give more reliable matching points than bilinear reparameterization method.

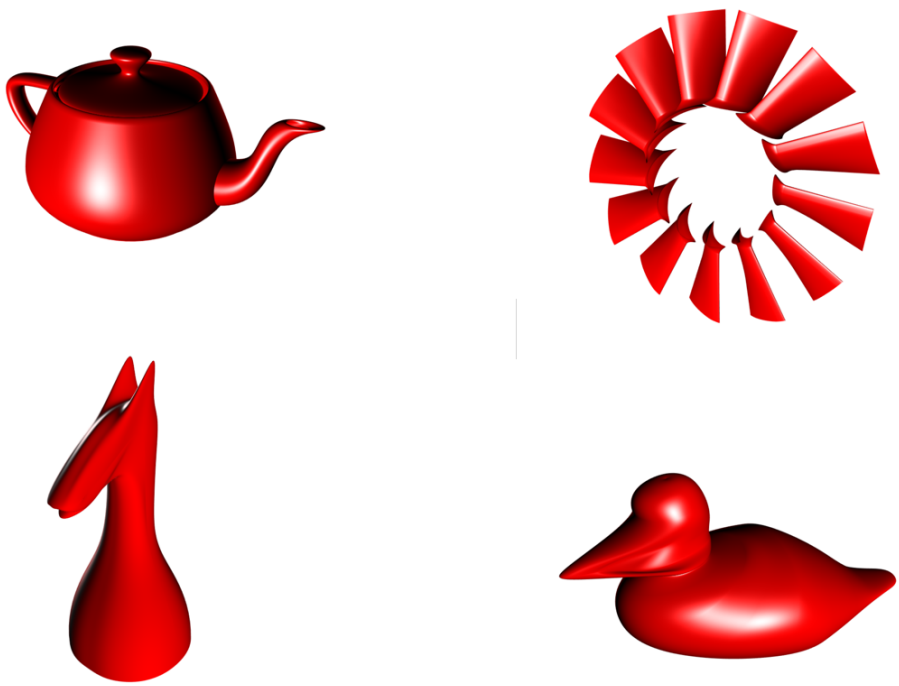


Figure 5.5 General parametric surface models : Teapot, Turbine, Knight and Duck

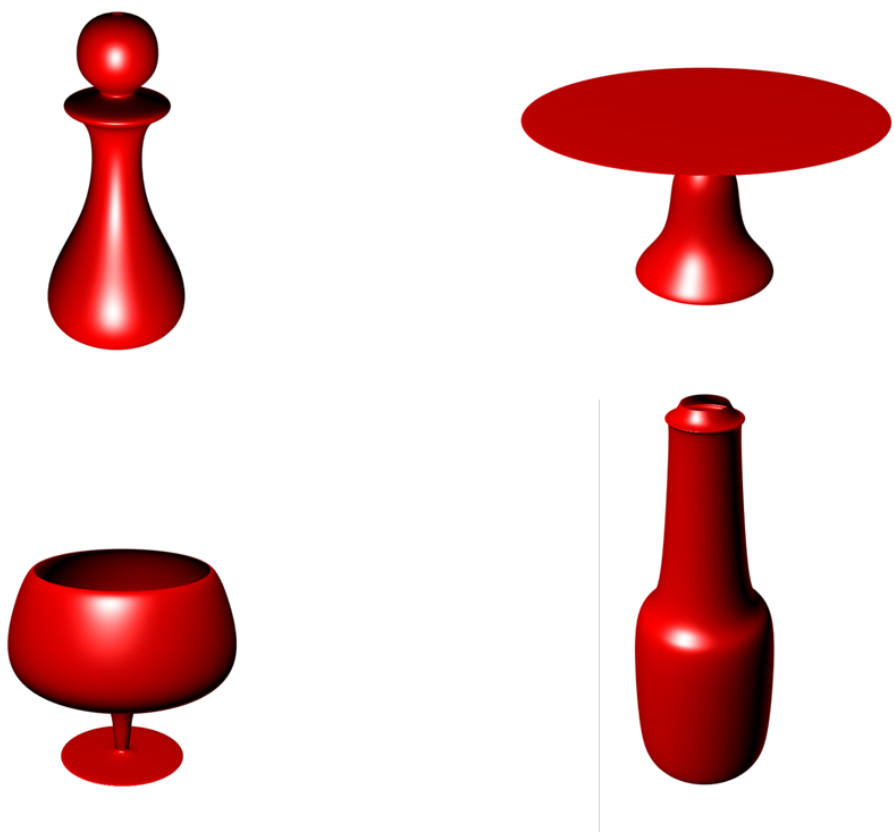


Figure 5.6 Surface of revolution models : Pawn, Table, Cup and Bottle

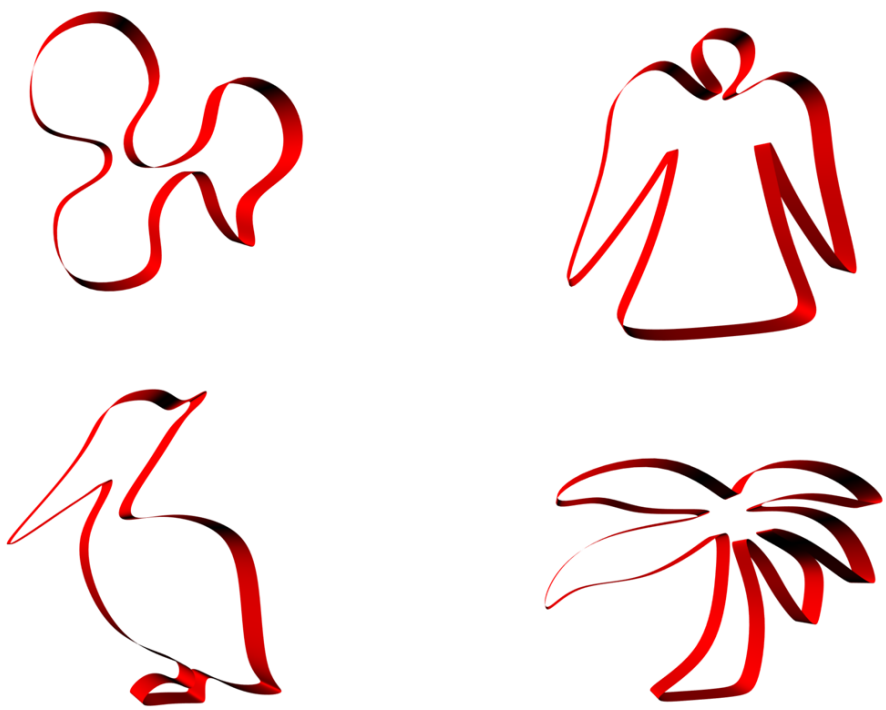


Figure 5.7 Surface of linear extrusion models : Duck, Angel, Bird and Tree

Teapot (Freeform), # Patch = 28										
	MEM 56GB					PREP 21 min				
	(a) MAX 146					(b) Max 98				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999965	0.0018397252	46	369	117	0.0013505127	0.0022217128	45	352	49
10^{-4}	0.0009999965	0.0010775719	69	581	73	0.0013505127	0.0014193708	60	492	14
10^{-5}	0.0009999965	0.0010012844	76	638	59	0.0013505127	0.0013592590	71	576	4
10^{-6}	0.0009999965	0.0010001217	76	641	58	0.0013505127	0.0013505633	71	580	1
10^{-7}	0.0009999965	0.0010000013	76	647	60	0.0013505127	0.0013505633	71	580	1
10^{-8}	0.0009999965	0.0010000013	78	652	59	0.0013505388	0.0013505485	82	796	15
10^{-9}	0.0009999998	0.0010000000	84	702	60	0.0013505388	0.0013505394	85	872	6
HD	0.00100000000000014		103	702	60	0.00135053902568166		85	872	6

Turbine (Freeform), # Patch = 156										
	MEM 49GB					PREP 10 min				
	(a) MAX 325					(b) Max 312				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0010004296	43	312	299	0.0018462048	0.0018467881	42	312	1
10^{-4}	0.0009999999	0.0010004296	43	312	299	0.0018462048	0.0018467881	42	312	1
10^{-5}	0.0009999999	0.0010004296	43	312	299	0.0018462048	0.0018467881	42	312	1
10^{-6}	0.0009999999	0.0010002877	43	312	299	0.0018462048	0.0018467881	42	312	1
10^{-7}	0.0009999999	0.0010000032	44	322	301	0.0018462048	0.0018462058	43	316	0
10^{-8}	0.0009999999	0.0010000000	44	322	301	0.0018462048	0.0018462058	43	316	0
10^{-9}	0.0009999999	0.0010000000	44	322	301	0.0018462048	0.0018462058	43	316	0
HD	0.00100000000000030		139	322	301	0.00184620513132141		43	316	0

Knight (Freeform), # Patch = 144										
	MEM 88GB					PREP 36 min				
	(a) MAX 529					(b) Max 288				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999229	0.0017349743	52	295	268	0.0028919001	0.0034223936	60	296	5
10^{-4}	0.0009999229	0.0010479521	53	308	268	0.0029948360	0.0030869721	60	316	3
10^{-5}	0.0009999229	0.0010033664	54	313	268	0.0029948360	0.0029951975	61	332	0
10^{-6}	0.0009999453	0.0010003913	58	330	272	0.0029948360	0.0029951975	61	332	0
10^{-7}	0.0009999731	0.0010000123	64	381	279	0.0029950301	0.0029951202	63	392	1
10^{-8}	0.0009999969	0.0010000010	117	712	254	0.0029950353	0.0029950394	64	416	1
10^{-9}	0.0009999996	0.0010000000	156	1702	271	0.0029950353	0.0029950354	64	424	0
HD	0.00100000000000056		291	1702	271	0.00299503547409318		64	424	0

Duck (Freeform), # Patch = 330										
	MEM 81GB					PREP 22 min				
	(a) MAX 768					(b) Max 660				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999304	0.0019431644	117	731	661	0.0023380711	0.0032987028	122	724	40
10^{-4}	0.0009999343	0.0010937231	152	906	661	0.0023380711	0.0024115996	126	780	16
10^{-5}	0.0009999343	0.0010078664	169	982	655	0.0023380711	0.0023446053	135	824	4
10^{-6}	0.0009999507	0.0010003818	179	1019	655	0.0023380711	0.0023382515	136	832	1
10^{-7}	0.0009999706	0.0010000226	197	1130	644	0.0023381176	0.0023382168	138	856	2
10^{-8}	0.0009999967	0.0010000017	263	1789	604	0.0023381334	0.0023381423	149	1076	5
10^{-9}	0.0009999994	0.0010000000	275	1942	604	0.0023381334	0.0023381335	149	1096	2
HD	0.00100000000000176		708	1942	604	0.00233813359828730		149	1096	2

Table 5.1 HD computation for two identical general freeform surfaces.

Pawn (Revolution), # Patch = 10										
	MEM 8 MB					PREP 43 msec				
	(a) MAX 54					(b) Max 52				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0010001177	17	104	47	0.0015926186	0.0015926188	12	104	0
10^{-4}	0.0009999999	0.0010001177	17	104	47	0.0015926186	0.0015926188	12	104	0
10^{-5}	0.0009999999	0.0010001177	17	104	47	0.0015926186	0.0015926188	12	104	0
10^{-6}	0.0009999999	0.0010001177	17	104	47	0.0015926186	0.0015926188	12	104	0
10^{-7}	0.0009999999	0.0010000464	19	107	47	0.0015926186	0.0015926188	12	104	0
10^{-8}	0.0009999999	0.0010000021	24	114	47	0.0015926186	0.0015926188	12	104	0
10^{-9}	0.0009999999	0.0010000001	29	118	47	0.0015926186	0.0015926188	12	104	0
HD	0.00100000000000005		33	118	47	0.00159261877683509		12	104	0

Table (Revolution), # Patch = 10										
	MEM 6 MB					PREP 32 msec				
	(a) MAX 42					(b) Max 42				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0011801595	13	84	38	0.0010322662	0.0019591548	15	90	3
10^{-4}	0.0009999999	0.0010237210	15	86	38	0.0010322662	0.0010990603	16	92	2
10^{-5}	0.0009999999	0.0010018032	16	87	37	0.0010322662	0.0010322663	17	96	0
10^{-6}	0.0009999999	0.0010000091	16	87	37	0.0010322662	0.0010322663	17	96	0
10^{-7}	0.0009999999	0.0010000091	16	87	37	0.0010322662	0.0010322663	17	96	0
10^{-8}	0.0009999999	0.0010000041	17	88	37	0.0010322662	0.0010322663	17	96	0
10^{-9}	0.0009999999	0.0010000000	19	91	36	0.0010322662	0.0010322663	17	96	0
HD	0.00100000000000001		23	91	36	0.00103226631971087		17	96	0

Cup (Revolution), # Patch = 10										
	MEM 8 MB					PREP 42 msec				
	(a) MAX 51					(b) Max 48				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0010931650	15	96	43	0.0010390303	0.0010495789	12	96	2
10^{-4}	0.0009999999	0.0010137794	15	96	43	0.0010390303	0.0010495789	12	96	2
10^{-5}	0.0009999999	0.0010062990	17	99	43	0.0010390303	0.0010484856	13	98	1
10^{-6}	0.0009999999	0.0010002437	20	100	41	0.0010390303	0.0010390305	15	100	0
10^{-7}	0.0009999999	0.0010000576	34	111	43	0.0010390303	0.0010390305	15	100	0
10^{-8}	0.0009999999	0.0010000035	43	124	42	0.0010390303	0.0010390305	15	100	0
10^{-9}	0.0009999999	0.0010000000	44	128	43	0.0010390303	0.0010390305	15	100	0
HD	0.00100000000000004		49	128	43	0.00103903045417239		15	100	0

Bottle (Revolution), # Patch = 10										
	MEM 7 MB					PREP 39 msec				
	(a) MAX 59					(b) Max 56				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0016393711	18	113	52	0.0018137399	0.0024329925	17	116	13
10^{-4}	0.0009999999	0.0010785444	27	124	48	0.0018137399	0.0018889304	32	136	3
10^{-5}	0.0009999999	0.0010010742	31	128	46	0.0018341133	0.0018342535	36	140	1
10^{-6}	0.0009999999	0.0010005046	32	129	47	0.0018341133	0.0018342535	35	140	1
10^{-7}	0.0009999999	0.0010000471	34	132	46	0.0018341257	0.0018341257	43	148	1
10^{-8}	0.0009999999	0.0010000014	48	144	47	0.0018341257	0.0018341257	43	148	1
10^{-9}	0.0009999999	0.0010000003	48	145	48	0.0018341257	0.0018341257	43	148	1
HD	0.00100000000000004		53	145	48	0.00183412577503611		43	148	1

Table 5.2 HD computation for two identical surfaces of revolution.

Duck (Extrusion), # Patch = 60										
	MEM 22 MB					PREP 127 msec				
	(a) MAX 541					(b) Max 214				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0010002405	21	214	210	0.0016847976	0.0017941514	20	214	16
10^{-4}	0.0009999999	0.0010002405	21	214	210	0.0016847976	0.0017816145	21	226	12
10^{-5}	0.0009999999	0.0010002405	21	214	210	0.0016856950	0.0016874449	25	246	1
10^{-6}	0.0009999999	0.0010002405	21	214	210	0.0016856950	0.0016856951	25	248	0
10^{-7}	0.0009999999	0.0010000869	30	268	228	0.0016856950	0.0016856951	25	248	0
10^{-8}	0.0009999999	0.0010000001	34	296	226	0.0016856950	0.0016856951	25	248	0
10^{-9}	0.0009999999	0.0010000001	34	296	226	0.0016856950	0.0016856951	25	248	0
HD	0.00100000000000004		50	296	226	0.00168569508447827		25	248	0

Angel (Extrusion), # Patch = 72										
	MEM 27 MB					PREP 155 msec				
	(a) MAX 547					(b) Max 246				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0010956826	24	246	235	0.0019580436	0.0022044477	23	246	5
10^{-4}	0.0009999999	0.0010089718	24	246	234	0.0019580436	0.0020216309	23	248	4
10^{-5}	0.0009999999	0.0010006311	24	247	234	0.0019610966	0.0019610968	26	258	0
10^{-6}	0.0009999999	0.0010002382	24	247	234	0.0019610966	0.0019610968	26	258	0
10^{-7}	0.0009999999	0.0010000796	30	287	243	0.0019610966	0.0019610968	26	258	0
10^{-8}	0.0009999999	0.0010000061	37	331	248	0.0019610966	0.0019610968	26	258	0
10^{-9}	0.0009999999	0.0010000005	39	340	249	0.0019610966	0.0019610968	26	258	0
HD	0.00100000000000005		60	340	249	0.00196109675504427		26	258	0

Bird (Extrusion), # Patch = 82										
	MEM 26 MB					PREP 164 msec				
	(a) MAX 462					(b) Max 298				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0010621390	29	298	266	0.0021825689	0.0022856429	29	298	7
10^{-4}	0.0009999999	0.0010133904	29	298	266	0.0022833798	0.0022833804	29	300	0
10^{-5}	0.0009999999	0.0010029761	29	298	266	0.0022833798	0.0022833804	29	300	0
10^{-6}	0.0009999999	0.0010001706	29	300	265	0.0022833798	0.0022833804	29	300	0
10^{-7}	0.0009999999	0.0010000903	31	306	264	0.0022833798	0.0022833804	29	300	0
10^{-8}	0.0009999999	0.0010000071	36	347	267	0.0022833798	0.0022833804	29	300	0
10^{-9}	0.0009999999	0.0010000001	39	359	266	0.0022833798	0.0022833804	29	300	0
HD	0.00100000000000005		56	359	266	0.00228338005164069		29	300	0

Tree (Extrusion), # Patch = 88										
	MEM 37 MB					PREP 208 msec				
	(a) MAX 815					(b) Max 318				
10^{-l}	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE	$\underline{h}$	$\overline{h}$	TIME	ITER	SIZE
10^{-3}	0.0009999999	0.0010373885	32	318	302	0.0017515875	0.0017788452	31	318	11
10^{-4}	0.0009999999	0.0010100370	32	318	302	0.0017515875	0.0017788452	31	318	11
10^{-5}	0.0009999999	0.0010017327	32	319	302	0.0017515875	0.0017614429	32	324	8
10^{-6}	0.0009999999	0.0010003937	32	320	302	0.0017515875	0.0017515875	35	340	0
10^{-7}	0.0009999999	0.0010000796	44	392	324	0.0017515875	0.0017515875	35	340	0
10^{-8}	0.0009999999	0.0010000067	52	441	322	0.0017515875	0.0017515875	35	340	0
10^{-9}	0.0009999999	0.0010000006	55	460	325	0.0017515875	0.0017515875	35	340	0
HD	0.00100000000000007		75	460	325	0.00175158752002347		35	340	0

Table 5.3 HD computation for two identical surfaces of linear extrusion.

Chapter 6

Conculsion

In this work, we have proposed to use toroidal patches to approximate various kinds of freeform parametric surfaces. For a general regular surface, we have suggested a reparameterization method for the toroidal patches that approximate the surface. For a surface of revolution and a surface of linear extrusion, we suggested to use circular arcs to approximate the profile curve and then generated toroidal, or cylindrical, patches from the circular arcs to approximate the surface.

Using these toroidal patches, we could construct BVH that is much more compact than the BVH that is based upon conventional convex bounding volumes. We also suggested two fundamental toroidal operations, point projection and binormal computation, that are applicable to our BVH. The toroidal point projection operation can be extended to point projection algorithm for parametric surface. Also, the toroidal binormal computation can be extended to various distance computation algorithms for parametric surface, as many dis-

tance measures occur on the binormal lines. Additionally, we discovered that we can find binormal lines of circles in space to find those of toroidal patches, which led to significant increase of efficiency.

Finally, we have shown that we can improve the minimum distance computation algorithm for solids of revolution, and the Hausdorff distance computation algorithm for various freeform parametric surfaces with our BVH. We could accelerate the minimum distance computation for solids of revolution, often 10-100 times faster. For the Hausdorff Distance computation algorithm, we improved both the speed and the precision of the algorithm. Especially, we enhanced the precision of the algorithm almost up to the machine precision, without significant increase of computational cost.

To sum up, we have dealt with both theoretical and practical aspect of toroidal patches in this work. We believe that we can do many more tasks with toroidal patches, as they must have many more useful geometric properties than we have suggested here. We hope this work will contribute to the future research that explore the possibilities of toroidal patches.

Appendices

Appendix A

Torus Reparameterization

If we create an osculating torus T to a regular surface S at the contact point $\mathbf{Q} = S(u_0, v_0) = T(s_0, t_0)$ by the procedure from Section 4.2.1, S and T have the same tangent plane at the point $\mathbf{Q}$. This observation leads to following lemma about the first derivatives of the surface $S(u, v)$ and the torus $T(s, t)$ at the point $\mathbf{Q}$.

Lemma 1. *There exist $\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{22}$ that satisfy following relationship :*

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \begin{bmatrix} T_s(s_0, t_0) \\ T_t(s_0, t_0) \end{bmatrix} = \begin{bmatrix} S_u(u_0, v_0) \\ S_v(u_0, v_0) \end{bmatrix}$$

Proof. Since the surface $S(u, v)$ is a regular surface, $S_u(u_0, v_0)$ and $S_v(u_0, v_0)$ are independent vectors that span the tangent plane $T_{S(u_0,v_0)}(S)$. Also, as we do not consider the cases where the point $\mathbf{Q}$ is parabolic or planar point, $T_s(s_0, t_0)$ and $T_t(s_0, t_0)$ are independent vectors that span the tangent plane $T_{T(s_0,t_0)}(T)$. Even when the principal curvatures κ_0 and κ_1 at the point Q

are the same non-zero values, the point Q becomes elliptic point and therefore $T_s(s_0, t_0)$ and $T_t(s_0, t_0)$ cannot be zero vectors. Since the two tangent planes are identical, there exist a linear transform that is expressed with $\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{22}$ that satisfy the above relationship. □

Note that the torus T degenerates to a cylinder if the point Q is a parabolic point and degenerates to a plane if the point Q is a planar point [36]. In these cases, we have to process them exceptionally.

Now, we propose a reparameterization method that computes corresponding torus parameter (s, t) for the given surface parameter (u, v). That is, we can approximate a point $S(u, v)$ on the regular surface with a point on the torus $T_2(u, v) := T(s(u, v), t(u, v))$.

Theorem 2. *Let $R \subset \mathbb{R}^2$ be a region that has neighborhood of (u_0, v_0). Let $S : R \to \mathbb{R}^3$ be any regular surface, and T be the torus approximation of S at the point $S(u_0, v_0)$ as stated above. Then, we can find the corresponding point $T(s, t)$ on the torus for the given point $S(u, v)$ on the surface with the derivatives of S, T and $\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{22}$ values from Lemma 1 :*

$$s(u, v) = c_1u^2 + c_2v^2 + c_3uv + c_4u + c_5v + c_6$$

$$t(u, v) = d_1u^2 + d_2v^2 + d_3uv + d_4u + d_5v + d_6$$

where

$$L_1 = S_{uu}(u_0, v_0) - (\alpha_{11}{}^2 T_{ss}(s_0, t_0) + 2\alpha_{11}\alpha_{12}T_{st}(s_0, t_0) + \alpha_{12}{}^2 T_{tt}(s_0, t_0))$$

$$L_2 = S_{vv}(u_0, v_0) - (\alpha_{21}{}^2 T_{ss}(s_0, t_0) + 2\alpha_{21}\alpha_{22}T_{st}(s_0, t_0) + \alpha_{22}{}^2 T_{tt}(s_0, t_0))$$

$$L_3 = S_{uv}(u_0, v_0) - (\alpha_{11}\alpha_{21}T_{ss}(s_0, t_0) + 2(\alpha_{11}\alpha_{22} + \alpha_{12}\alpha_{21})T_{st}(s_0, t_0) + \alpha_{12}\alpha_{22}T_{tt}(s_0, t_0))$$

and

$$c_1 = \frac{\langle L_1, T_s(s_0,t_0)\rangle}{2\|T_s(s_0,t_0)\|^2}$$
$$c_2 = \frac{\langle L_2, T_s(s_0,t_0)\rangle}{2\|T_s(s_0,t_0)\|^2}$$
$$c_3 = \frac{\langle L_3, T_s(s_0,t_0)\rangle}{\|T_s(s_0,t_0)\|^2}$$
$$c_4 = \alpha_{11} - 2c_1u_0 - c_3v_0$$
$$c_5 = \alpha_{21} - 2c_2v_0 - c_3u_0$$
$$c_6 = s_0 - c_1{u_0}^2 - c_2{v_0}^2 - c_3u_0v_0 - c_4u_0 - c_5v_0$$

and

$$d_1 = \frac{\langle L_1, T_t(s_0,t_0)\rangle}{2\|T_t(s_0,t_0)\|^2}$$
$$d_2 = \frac{\langle L_2, T_t(s_0,t_0)\rangle}{2\|T_t(s_0,t_0)\|^2}$$
$$d_3 = \frac{\langle L_3, T_t(s_0,t_0)\rangle}{\|T_t(s_0,t_0)\|^2}$$
$$d_4 = \alpha_{12} - 2d_1u_0 - d_3v_0$$
$$d_5 = \alpha_{22} - 2d_2v_0 - d_3u_0$$
$$d_6 = t_0 - d_1{u_0}^2 - d_2{v_0}^2 - d_3u_0v_0 - d_4u_0 - d_5v_0$$

Then, the torus $T_2(u,v) := T(s(u,v),t(u,v))$ *has the identical first and second derivatives with the surface* $S(u,v)$ *at* (u_0,v_0) *as follows :*

$$S_u(u_0,v_0) = T_{2,u}(u_0,v_0)$$
$$S_v(u_0,v_0) = T_{2,u}(u_0,v_0)$$
$$S_{uu}(u_0,v_0) = T_{2,uu}(u_0,v_0)$$
$$S_{uv}(u_0,v_0) = T_{2,uv}(u_0,v_0)$$
$$S_{vv}(u_0,v_0) = T_{2,vv}(u_0,v_0)$$

Proof. Since we can directly prove this theorem by calculating the first and second derivatives of $S(u,v)$ and $T_2(u,v)$ at (u_0,v_0), we omit the detailed proof. □

When we use this reparameterization method, we can also calculate the upper bound of approximation error $\|S(u,v)-T_2(u,v)\|$ for a specific domain as follows.

Theorem 3. *Let $R \subset \mathbb{R}^2$ be the rectangular region with vertices A,B,C,D of the form $A=(u_0,v_0), B=A+(l_1,0), C=A+(l_1,l_2), D=A+(0,l_2)$. Let $S: R \to \mathbb{R}^3$ be any regular surface, and $T_2(u,v) := T(s(u,v),t(u,v))$ be the torus approximation of S at the point $S(u_0,v_0)$ as stated above. Then*

$$\max_{(u,v)\in R}\|S(u,v)-T_2(u,v)\| \leq$$

$$\frac{1}{6}({l_1}^3(M1+N1)+3{l_1}^2l_2(M2+N2)+3l_1{l_2}^2(M3+N3)+{l_2}^3(M4+N4))$$

where

$$M_1=\max_{(u,v)\in R}\|S_{uuu}(u,v)\|, N_1=\max_{(u,v)\in R}\|T_{2,uuu}(u,v)\|$$

$$M_2=\max_{(u,v)\in R}\|S_{uuv}(u,v)\|, N_2=\max_{(u,v)\in R}\|T_{2,uuv}(u,v)\|$$

$$M_3=\max_{(u,v)\in R}\|S_{uvv}(u,v)\|, N_3=\max_{(u,v)\in R}\|T_{2,uvv}(u,v)\|$$

$$M_4=\max_{(u,v)\in R}\|S_{vvv}(u,v)\|, N_4=\max_{(u,v)\in R}\|T_{2,vvv}(u,v)\|$$

Proof. Let

$$P_0=(u_0,v_0)\in R$$

$$\mathbf{e}(u,v)=S(u,v)-T_2(u,v).$$

Then, from Theorem 2,

$$\mathbf{e}(P_0) = \mathbf{e_u}(P_0) = \mathbf{e_v}(P_0) = \mathbf{e_{uu}}(P_0) = \mathbf{e_{uv}}(P_0) = \mathbf{e_{vv}}(P_0) = 0$$

Now, let

$$P_1 = (u_0, v_1) \in R$$

$$P_2 = (u_1, v_1) \in R$$

$$\mathbf{v} = P_2 - P_1 = (u_1 - u_0, 0)$$

$$\mathbf{q}(t) = \mathbf{e}(P_1 + t\mathbf{v}).$$

Then

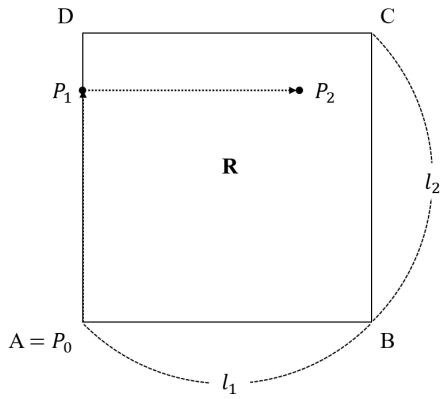
$$\mathbf{q}'(t) = \mathbf{v}_0 \mathbf{e}_u(P_1 + t\mathbf{v})$$

$$\mathbf{q}''(t) = {\mathbf{v}_0}^2 \mathbf{e}_{uu}(P_1 + t\mathbf{v})$$

$$\mathbf{q}'''(t) = {\mathbf{v}_0}^3 \mathbf{e}_{uuu}(P_1 + t\mathbf{v}) \Rightarrow \max_{0 \le t \le 1} \|\mathbf{q}'''(t)\| \le {l_1}^3 (M1 + N1)$$

Now, we calculate the upper bound of $\|\mathbf{q}(0)\|$. Let

$$\mathbf{w} = P_1 - P_0 = (0, v_1 - v_0)$$



$$\mathbf{r}(t) = \mathbf{e}(P_0 + t\mathbf{w}).$$

Then

$$\mathbf{r}(0) = \mathbf{e}(P_0) = 0$$

$$\mathbf{r}'(t) = \mathbf{w}_1\mathbf{e}_v(P_0 + t\mathbf{w}) \Rightarrow \mathbf{r}'(0) = \mathbf{w}_1\mathbf{e}_v(P_0) = 0$$

$$\mathbf{r}''(t) = {\mathbf{w}_1}^2\mathbf{e}_{vv}(P_0 + t\mathbf{w}) \Rightarrow \mathbf{r}''(0) = {\mathbf{w}_1}^2\mathbf{e}_{vv}(P_0) = 0$$

$$\mathbf{r}'''(t) = {\mathbf{w}_1}^3\mathbf{e}_{vvv}(P_0 + t\mathbf{w}) \Rightarrow \max_{0\le t\le 1} \|\mathbf{r}'''(t)\| \le {l_2}^3(M4 + N4).$$

By Taylor expansion, we get

$$\begin{aligned}\mathbf{r}(t) &= \mathbf{r}(0) + \mathbf{r}'(0)t + \frac{1}{2}\mathbf{r}''(0)t^2 + \int_0^t \mathbf{r}'''(\xi)\frac{(t-\xi)^2}{2}d\xi \\ &= \int_0^t \mathbf{r}'''(\xi)\frac{(t-\xi)^2}{2}d\xi\end{aligned}$$

Then

$$\mathbf{r}(1) = \int_0^1 \mathbf{r}'''(\xi)\frac{(1-\xi)^2}{2}d\xi$$

and

$$\begin{aligned}\|\mathbf{q}(0)\| = \|\mathbf{r}(1)\| &\le \max_{0\le t\le 1} \|\mathbf{r}'''(t)\| \int_0^1 \frac{(1-\xi)^2}{2}d\xi \\ &\le \frac{1}{6}{l_2}^3(M4 + N4).\end{aligned}$$

Likewise, we can calculate the upper bound of $\|\mathbf{q}'(0)\|, \|\mathbf{q}''(0)\|$:

$$\|\mathbf{q}'(0)\| \le \frac{1}{2}l_1{l_2}^2(M3 + N3)$$

$$\|\mathbf{q}''(0)\| \le {l_1}^2 l_2(M2 + N2).$$

Now, by Taylor expansion of $\mathbf{q}(t)$, we get

$$\mathbf{q}(t) = \mathbf{q}(0) + \mathbf{q}'(0)t + \frac{1}{2}\mathbf{q}''(0)t^2 + \int_0^t \mathbf{q}'''(\xi)\frac{(t-\xi)^2}{2}d\xi$$

$$\Longrightarrow \mathbf{q}(1) = \mathbf{q}(0) + \mathbf{q}'(0) + \frac{1}{2}\mathbf{q}''(0) + \int_0^1 \mathbf{q}'''(\xi)\frac{(1-\xi)^2}{2}d\xi$$

which leads to

$$\|\mathbf{q}(1)\| \leq \|\mathbf{q}(0)\| + \|\mathbf{q}'(0)\| + \frac{1}{2}\|\mathbf{q}''(0)\| + \max_{0\leq\xi\leq 1}\|\mathbf{q}'''(\xi)\|\|\int_0^1 \frac{(1-\xi)^2}{2}d\xi\|$$

and thus

$$\|\mathbf{e}(u_1, v_1)\| = \|\mathbf{q}(1)\| \leq$$

$$\frac{1}{6}({l_1}^3(M1+N1) + 3{l_1}^2 l_2(M2+N2) + 3l_1{l_2}^2(M3+N3) + {l_2}^3(M4+N4))$$

Since $P_1 \in R$ is an arbitrary parameter in R, the theorem holds. □

Bibliography

[1] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of computational geometry*, pages 121–153. Elsevier, 2000.

[2] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proceedings. IEEE international conference on multimedia and expo*, volume 1, pages 705–708. IEEE, 2002.

[3] M. Bartoň, I. Hanniel, G. Elber, and M.-S. Kim. Precise hausdorff distance computation between polygonal meshes. *Computer Aided Geometric Design*, 27(8):580–591, 2010.

[4] G. v. d. Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of graphics tools*, 2(4):1–13, 1997.

[5] P. Bo and M. Bartoň. 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.

[6] G. Bradshaw and C. O'Sullivan. Sphere-tree construction using dynamic medial axis approximation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 33–40, 2002.

[7] S. Cameron. Enhancing gjk: Computing minimum and penetration distances between convex polyhedra. In *Proceedings of international conference on robotics and automation*, volume 4, pages 3112–3117. IEEE, 1997.

[8] 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.

[9] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. In *Computer graphics forum*, volume 17, pages 167–174. Wiley Online Library, 1998.

[10] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[11] M. P. Do Carmo. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.

[12] D. Eberly. *3D game engine design: a practical approach to real-time computer graphics*. CRC Press, 2006.

[13] G. Elber and T. Grandine. Hausdorff and minimal distances between parametric freeforms in $\mathbb{R}^2$ and $\mathbb{R}^3$. In *International Conference on Geometric Modeling and Processing*, pages 191–204. Springer, 2008.

[14] G. Farin and D. Hansford. *The essentials of CAGD*. CRC Press, 2000.

[15] R. T. Farouki. The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, 2012.

[16] D. Filip, R. Magedson, and R. Markot. Surface algorithms using bounds on derivatives. *Computer Aided Geometric Design*, 3(4):295–311, 1986.

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

[18] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, 1996.

[19] I. Hanniel, A. Krishnamurthy, and S. McMains. Computing the hausdorff distance between nurbs surfaces using numerical iteration on the gpu. *Graphical Models*, 74(4):255–264, 2012.

[20] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics (TOG)*, 15(3):179–210, 1996.

[21] B. Jiittler. Bounding the hausdorff distance of implicitly defined and/or parametric curves. 2001.

[22] D. E. Johnson and E. Cohen. A framework for efficient minimum distance computations. In *Proceedings. 1998 IEEE International Conference on*

Robotics and Automation (Cat. No. 98CH36146), volume 4, pages 3678–3684. IEEE, 1998.

[23] D. E. Johnson and E. Cohen. *Minimum distance queries for haptic rendering*. PhD thesis, School of Computing, University of Utah, 2005.

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

[25] 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.

[26] 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 (TOG)*, 30(6):1–8, 2011.

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

[28] J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[29] A. Krishnamurthy, S. McMains, and I. Hanniel. Gpu-accelerated hausdorff distance computation between dynamic deformable nurbs surfaces. *Computer-Aided Design*, 43(11):1370–1379, 2011.

[30] S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. Rapid and accurate contact determination between spline models using shelltrees. In *Computer Graphics Forum*, volume 17, pages 315–326. Wiley Online Library, 1998.

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

[32] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical report, Technical Report TR99-018, Department of Computer Science, University of . . . , 1999.

[33] 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.

[34] 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.

[35] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *ICRA*, volume 91, pages 9–12, 1991.

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

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

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

[39] C. A. Neff. Finding the distance between two circles in three-dimensional space. *IBM journal of research and development*, 34(5):770–775, 1990.

[40] Y. Park, S.-H. Son, M.-S. Kim, and G. Elber. Surface-surface-intersection computation using a bounding volume hierarchy with osculating toroidal patches in the leaf nodes. *Computer-Aided Design*, page 102866, 2020.

[41] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 1996.

[42] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical recipes in c, 1988.

[43] S. Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3324–3329. IEEE, 1994.

[44] L. Scharf. Computing the hausdorff distance between sets of curves. *Freie Universitat Berlin*, 2003.

[45] 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.

[46] Z. Šír, R. Feichtinger, and B. Jüttler. Approximating curves and their offsets using biarcs and pythagorean hodograph quintics. *Computer-Aided Design*, 38(6):608–618, 2006.

[47] R. Smith et al. Open dynamics engine. 2005.

[48] S.-H. Son, S.-H. Yoon, M.-S. Kim, and G. Elber. Efficient minimum distance computation for solids of revolution. In *Computer Graphics Forum*, volume 39, pages 535–544. Wiley Online Library, 2020.

초 록

본 논문에서는 토러스 패치를 이용한 효율적인 기하학적 알고리즘들을 소개한다. 먼저, 임의의 일반적인 정칙 곡면을 근사하기 위해 최대 접촉 토러스 패치를 사용할 것을 제안한다. 이를 위해 정칙 곡면의 변수를 토러스 패치의 변수로 변환하는 재매개화 공식을 제시한다. 이에 더해, 토러스 패치가 평면 곡선에 기반한 특수한 곡면들을 일반 곡면들보다 더 효과적으로 근사할 수 있음을 보인다.

이러한 토러스 패치의 정확성 덕분에, 임의의 곡면을 감싸는 굉장히 효율적인 bounding volume hierarchy를 얻을 수 있다. 이 자료 구조를 이용하여 공간 상의 한 점에서 해당 곡면으로 점 투영 연산을 굉장히 빠르고 정확하게 할 수 있다. 또한, 곡면들 사이의 다양한 거리들을 찾기 위해 이 자료 구조에 저장된 토러스 패치들을 수직으로 연결하는 binormal 직선을 이용할 수 있다. 이러한 binormal 직선을 효율적으로 찾기 위해 공간 상의 원들을 이용할 수 있음을 보인다.

토러스 패치가 제공하는 위와 같은 기초적인 기하학적 연산들을 토대로, 효율적인 회전체 사이의 최단 거리 계산 알고리즘을 제시한다. 이 알고리즘은 기존의 알고리즘에 비해 10-100배 빠른 속도로 최단 거리를 계산한다. 또한, 효율적인 하우스도르프 거리 계산 알고리즘 역시 제안한다. 실험 결과, 이 알고리즘을 통해 거의 기계 정확도 내에서 정확한 하우스도르프 거리를 큰 비용 증가 없이 계산할 수 있었다. 이와 같은 성능 향상은 본 논문에서 사용한 토러스 패치의 정확성과 효율성에 기반하고 있다.

주요어: 토러스 패치, Bounding Volume Hierarchy, 점 투영, Binormal 계산, 최단 거리 계산, 하우스도르프 거리 계산

학번: 2019-27608

Acknowledgments