공학박사학위논문

Trimming Self-Intersections of Offset Curves and Surfaces

오프셋 곡선 및 곡면의 자가 교차 검출 및 제거

2020 년 2 월

서울대학교 대학원

컴퓨터공학부

홍 규 연

오프셋 곡선 및 곡면의 자가 교차 검출 및 제거

Trimming Self-Intersections of Offset Curves and Surfaces

지도교수 김 명 수

이 논문을 공학박사 학위논문으로 제출함

2020 년 1 월

서울대학교 대학원

컴퓨터 공학부

홍 규 연

홍규연의 공학박사 학위논문을 인준함

2020 년 1 월

위 원 장	신영길	(인)
부위원장	김명수	(인)
위 원	이제희	(인)
위 원	경민호	(인)
위 원	윤승현	(인)

Abstract

Offset curves and surfaces have many applications in computer-aided design and manufacturing, but the self-intersections and redundancies must be trimmed away for their practical use. We present a new method for offset curve and surface trimming that detects the self-intersections and eliminates the redundant parts of an offset curve and surface that are closer than the offset distance to the original curve and surface.

We first propose an offset trimming method based on constructing geometric constraint equations. We formulate the constraint equations of the self-intersections of an offset curve and surface in the parameter domain of the original curve and surface. Numerical computations based on the regularity and intrinsic properties of the given input curve and surface is carried out to compute the solution of the constraint equations. The method deals with numerical instability around near-singular regions of an offset surface by using osculating tori that can be constructed in a highly stable way, i.e., by offsetting the osculating torii of the given input regular surface. We reveal the branching structure and the terminal points from the complete self-intersection curves of the offset surface.

From the observation that the trimming method based on the multivariate equation solving is computationally expensive, we also propose an acceleration technique to trim an offset curve and surface. The alternative method constructs a bounding volume hierarchy specially designed to enclose the offset curve and surface and detects the self-collision of the bounding volumes

instead. In the case of an offset surface, the thickness of the bounding volumes is indirectly determined based on the maximum deviations of the positions and the normals between the given input surface patches and their osculating tori. For further acceleration, the bounding volumes are pruned as much as possible during self-collision detection using various geometric constraints imposed on the offset surface. We demonstrate the effectiveness of the new trimming method using several non-trivial test examples of offset trimming.

Lastly, we investigate the problem of computing the Voronoi diagram of a freeform surface using the offset trimming technique for surfaces. By trimming the offset surface with a gradually changing offset radius, we compute the boundary of the Voronoi cells that appear in the concave side of the given input surface. In particular, we interpret the singular and branching points of the self-intersection curves of the trimmed offset surfaces in terms of the boundary elements of the Voronoi diagram.

Keywords: Offset Curve, Offset Surface, Offset Self-intersection, Trimming Redundancies, Near-singular Regions, Branching Structure, Bounding Volume Hierarchy, Voronoi Diagram

Student Number: 2016-30283

Contents

Abstract I

Contents **III**

List of Figures **VII**

List of Tables **XIII**

Chapter 1 Introduction **1**

1.1 Background and Motivation 1

1.2 Research Objectives and Approach 7

1.3 Contributions and Thesis Organization 11

Chapter 2 Preliminaries **14**

2.1 Curve and Surface Representation 14

2.1.1 Bézier Representation 14

2.1.2 B-spline Representation 17

2.2 Differential Geometry of Curves and Surfaces 19

2.2.1 Differential Geometry of Curves 19

2.2.2 Differential Geometry of Surfaces 21

Chapter 3 Previous Work **23**

3.1 Offset Curves 24

3.2 Offset Surfaces 27

3.3 Offset Curves on Surfaces 29

Chapter 4 Trimming Offset Curve Self-intersections **32**

4.1 Experimental Results 35

Chapter 5 Trimming Offset Surface Self-intersections **38**

5.1 Constraint Equations for Offset Self-Intersections 38

5.1.1 Coplanarity Constraint 39

5.1.2 Equi-angle Constraints 40

5.2 Removing Trivial Solutions 40

5.3 Removing Normal Flips 41

5.4 Multivariate Solver for Constraints 43

5.A Derivation of $f(u, v)$ 46

5.B Relationship between $f(u, v)$ and Curvatures 47

5.3 Trimming Offset Surfaces 50

5.4 Experimental Results 53

5.5 Summary 57

Chapter 6 Acceleration of trimming offset curves and surfaces **62**

6.1 Motivation 62

6.2 Basic Approach 67

6.3 Trimming an Offset Curve using the BVH 70

6.4 Trimming an Offset Surface using the BVH 75

6.4.1 Offset Surface BVH 75

6.4.2 Finding Self-intersections in Offset Surface Using BVH . 87
6.4.3 Tracing Self-intersection Curves 98
6.5 Experimental Results 100
6.6 Summary 106

Chapter 7 Application of Trimmed Offset Surfaces: 3D Voronoi Diagram **107**
7.1 Background 107
7.2 Approach 110
7.3 Experimental Results 112
7.4 Summary 114

Chapter 8 Conclusion **119**

Bibliography **i**
초록 xiii
감사의 글 xv

List of Figures

Figure 1.1 Offsetting applications. (a), (b) Sculpting the inner/outer surface of sheet-like objects by offsetting their outer/inner counterparts. (c) Planning a path of the ball cutter in NC machining. (d) Offset path of the ball cutter avoids overcutting and undercutting in NC machining (image in (a) from [40] and images in (c), (d) from [13]). . . . 3

Figure 1.2 (a) Offset curve formulation. (b) Offset surface and its progenitor surface. Singularity of offset surface is drawn in the pink circle. 5

Figure 4.1 The eqilateral triangle relation at offset curve self-intersection. 34

Figure 4.2 Trimmed offset curves. The progenitor curves are shown in black. 37

Figure 5.1 The eqilaternal triangle from an offset self-intersection condition. 41

Figure 5.2 Curvature analysis on the surface $S(u, v)$: (a) the outer loop (in black) is the boundary of a redundant trimming region, whereas the red region is the set of (u, v)-parameters where $S(u, v)$ has one of the principal curvatures to the concave side larger than $1/d$; (b) a zoom-in view on the upper-right corner of the region. 44

Figure 5.3 (a) An arrangement of solution curve segments from the constraint solver in the uv-domain; (b) the same curve arrangement, where matching segments are in the same color and matching endpoints are shown in blue line connections; (c) the arrangement of self-intersection curve segments in the xyz-space, where the color shows the correspondence with the matching uv-curve segments; (d) an X-junction with four branches of trimmed self-intersection curve segments on the offset surface (in the same color coding with (b) and (c)). 45

Figure 5.4 Examples from Maekawa et al. [62]: (a) the arrangement of solution curves in the uv-domain; (b) the input surface $S(u, v)$ (in blue) and the offset surface $O(u, v)$ (in red), and the offset trimming curve (in black) in the xyz-space; (c) zoom-in view of (b). 55

Figure 5.5 1st row: x:y = 1:1, 2nd row: x:y = 1.01:1, 3nd row: x:y = 1.05:1, 4th row: x:y = 1.2:1, 5th row: x:y = 1.5:1, 6th row: x:y = 2:1. (a) Input surface $S(u, v)$; (b) offset surface $O(u, v)$; (c), (d) trimmed self-intersection curves on the offset surface (from (c): side-view and (d) bottom-view). 58

Figure 5.6 Zoom-in views of the offset trimming curves on the offset surface $O(u, v)$ in the xyz-space, where the control nets are scaled along the x-direction in the ratios of $x : y =$ (a) $1 : 1$, (b) $1.01 : 1$, (c) $1 : 05 : 1$, (d) $1.2 : 1$, (e) $1.5 : 1$, and (f) $2 : 1$, where the offset distance is fixed to $d = 0.15$. Cyan points in (e) and (f) represent detected singular points where $O(u, v) = O(s, t)$. 59

Figure 5.7 Offset trimming curves: (a) in the uv-domain and (b), (c) in the xyz-space; (d)–(i) zoom-in views of the offset trimming curves (when viewed from below toward the upward direction) on the concave side of the offset surfaces (with the offset radius $d =$ (d) 0.1575, (e) 0.1545, (f) 0.1515, (g) 0.1485, (h) 0.1455, (i) 0.1200). 60

Figure 5.8 Examples from Seong et al. [78]: (a): the arrangement of solution curves in the uv-domain; (b): the input surface $S(u, v)$ (in blue) and the offset surface $O(u, v)$ (in red), and the offset trimming curve (in black) in the xyz-space. 61

Figure 6.1 Examples of trimmed offset curves. 74

Figure 6.2 Bounding volumes enclosing offset curves. Axis-aligned bounding boxes(AABBs) are used to bound monotone offset curve segments. 74

Figure 6.3 (a) An example offset tetrahedron enclosing the surface. (b) Recursive subdivion of the domain in building the BVH of the surface. 76

Figure 6.4 (a) Oscualting tori at the progenitor surface. (b) Offset of torus is also the osculating torus of the offset surface. 4 adjacent toroidal patches of progenitor surface (c) and their counterparts on offset surface (d). 81

Figure 6.5 Schematic diagram of the bounding error computation using the offset osculating tori. 83

Figure 6.6 Sample points to compute ϵ_{offset_torus} on (a) the outer torus and (b) the inner torus patches. 86

Figure 6.7 Normal-flipped regions of the offset surface (a) in the xyz-space and (b) the zoom-in view of (a); the corresponding region (c) in the uv-domain and (d) the zoom-in view of (c). 92

Figure 6.8 The common parent nodes (in red) of two leaf bounding volumes (in black) in (a) the originial bvh and (b) the bvh with the additional overlapping nodes. 96

Figure 6.9 (a) Intersection curves from numerical tracing. (b) The zoom-in view of the green circle marked on (a). 100

Figure 6.10 The colliding bounding volumes (in red) in the uv-domain as a result of acceleration techniques. 101

Figure 6.11 Collision pairs in xyz-domain as a result of acceleration techniques. 103

Figure 7.1 A relation between trimmed offset curves with varying offset distances (left) and a Voronoi diagram of a curve (right) (images from Held [35]). 109

Figure 7.2 Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space. 112

Figure 7.3 Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space. 116

Figure 7.4 Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space. 117

Figure 7.5 Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space. 118

Figure 7.6 Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space. 118

List of Tables

Table 6.1 Execution time of trimming offset surfaces based on the multivariate equation solving. d denotes the offset distance. 65

Table 6.2 The impact of the degree of the progenitor surfaces on the performace of offset trimming algorithm. 66

Table 6.3 The number of bounding volume pairs in collision. . . . 102

Table 6.4 Execution time (in hh:mm:ss) of trimming algorithms using the multivariate equation solving (third column) and the BVH-baed acceleration techniques (in fourth column). 105

Chapter 1

Introduction

1.1 Background and Motivation

Mass production of standardized industrial products in machine-controlled assembly lines is the gist of the modern manufacturing system. Unlike traditional craft production, it is important to describe what and how to produce in a precise language. The area of computer-aided design and manufacturing (CAD/CAM) emerged in the 1950s, and since then has been extensively used in numerically controlled machining in the automotive and aircraft industries [23]. A description has been an essential tool for a rapid and controllable production process.

In the applications of CAD/CAM, freeform curves, surfaces, and solids have been widely used as primitives to represent the shape of industrial objects. They are defined in precise mathematical terms and the characteristics have been studied in a prolific context in mathematics for thousands of years. In the

early ages of CAD/CAM, the main focus was on the representation of these geometric entities and how to design these geometric objects in numerically compact and efficient ways. Research interest was since then expanded to analyzing various properties of curves, surfaces and solids and manipulating the designed objects to sculpt new objects. *Geometry Processing*, the term first coined by Barnhill and Riesenfeld [6, 68], is a subfield of CAD/CAM that mainly focuses on the analysis of geometric properties of curves, surfaces and solids. Examples of geometry processing include the curvature analysis of curves and surfaces, contour extraction, the computation of offsets of curves and surfaces, and so forth.

The development of geometry processing has introduced many useful geometric analysis tools. *Offsetting* is one of these essential tools in the CAD/CAM applications. Offset operation in CAD/CAM reproduces new curves and surfaces with the constant spacing from the original curves, surfaces and solids. Offsetting is an essential component in CAD/CAM systems because it is much more cost-effective to alter the existing object than to create the new shape of an object from scratch. In the automotive or aircraft industry, for instance, the hull of a car or a plane is sculpted from a set of surfaces. The thickness of surfaces, however, might vary depending on the choice of the material or the detailed design. To fabricate the steel plate from the designed surface, the surface must be offset to a real-world object having the given thickness.

Offsetting is also essential in designing a way of manufacturing industrial objects. Modern factories often utilize numerically controlled (NC) machining to produce objects when the precise measurement is required. In NC machining, an object is finished by subtracting a raw material with cutter tools that have various shapes of tips attached. To sculpt the correct shape and avoid a

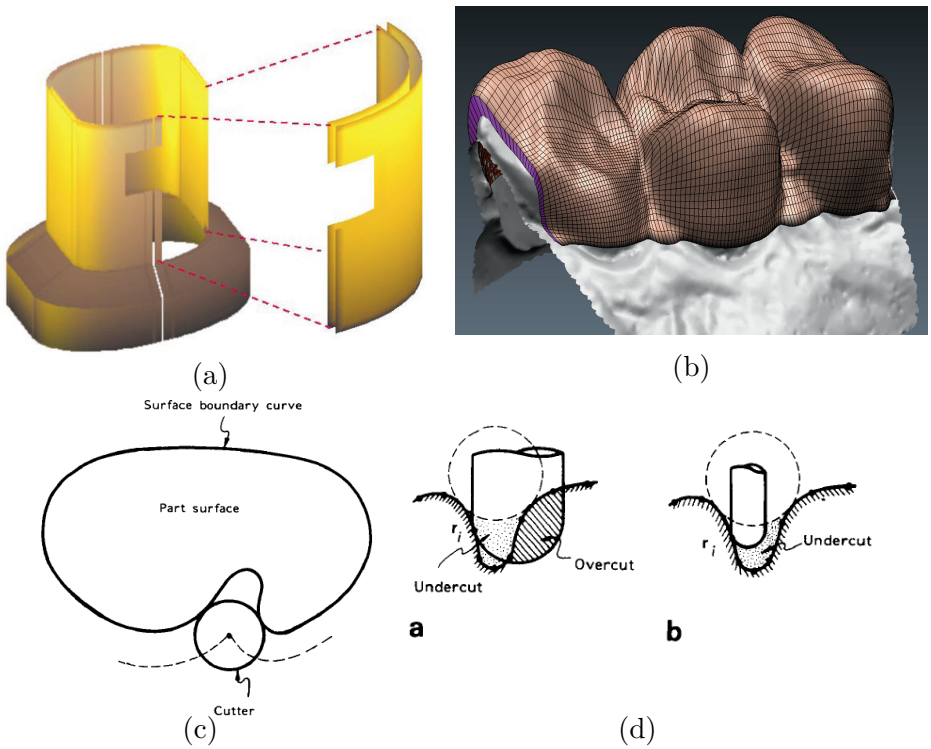


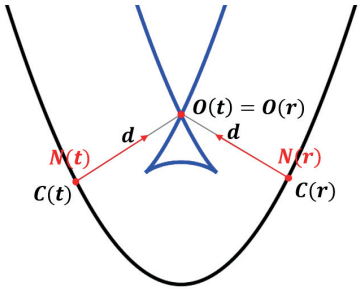
Figure 1.1: Offsetting applications. (a), (b) Sculpting the inner/outer surface of sheet-like objects by offsetting their outer/inner counterparts. (c) Planning a path of the ball cutter in NC machining. (d) Offset path of the ball cutter avoids overcutting and undercutting in NC machining (image in (a) from [40] and images in (c), (d) from [13]).

waste of materials, the paths of the cutter tools must follow the surface of the design, but the size and the shape of the cutter tips must be considered when planning the path. If it fails to plan the cutter path correctly, the cutter will overcut or undercut the hull of the designed object, as shown in Figure 1.1 (d). As explained above, offsetting curves and surfaces has been one of the most fundamental components of geometry processing tools with a wide variety

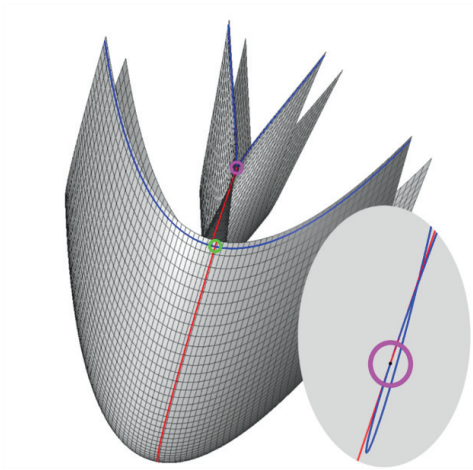
of CAD/CAM applications including tool path generation for CNC machining [11, 92], tolerance analysis [67], collision-free access space representation in robotics, brush stroke representation [43], and so on [63, 72, 74].

Theoretically speaking, an offset curve or surface (also known as a parallel curve or surface in geometry) is a curve or surface of which loci have a constant distance d from its progenitor curve or surface [80]. In the majority of its applications, the distance is taken along the normal direction of the curve or surface so that the offset curve or surface would have constant *thickness* as a result. Despite its significance and bountiful applications, offsetting has several research challenges to be addressed. First of all, offset curves and surfaces tend to show more sophisticated and pathological behavior than their progenitor curves and surfaces. This is because offset curves and surfaces often do not belong to the same function class as the progenitor curves and surfaces. For instance, curves and surfaces are represented as parametrized functions in many CAD/CAM applications. In particular, rational polynomial functions such as rational Bézier functions and rational B-spline functions are dominantly used in CAD/CAM as they have a compact representation while offering great controllability and numerical stability (Farin [23]). Nevertheless, the offsets of rational curves and surfaces become non-rational in general, which causes lots of technical problems.

The non-rationality of offsets of rational curves and surfaces can be easily confirmed by checking the mathematical equations of offset curves and surfaces. Given a regular parametric curve $C(t)$ and a regular parametric surface $S(u, v)$, the offset curve $O(t)$ and the offset surface $O(u, v)$ of distance $d > 0$



(a)



(b)

Figure 1.2: (a) Offset curve formulation. (b) Offset surface and its progenitor surface. Singularity of offset surface is drawn in the pink circle.

are defined as follows:

$$O(t) = C(t) + d \cdot N(t) \tag{1.1}$$

$$O(u, v) = S(u, v) + d \cdot N(u, v) \tag{1.2}$$

where $N(t)$ and $N(u, v)$ is a unit normal field of $C(t)$ and $S(u, v)$, respectively. Because of the normalized terms $N(t)$ and $N(u, v)$ in their formulation, $O(t)$ and $O(u, v)$ are usually non-rational.

Another challenge of offset curves and surfaces arises from the fact that the regularity of the progenitor curves and surfaces do not guarantee the regularity of their offsets any more. One can observe this phenomenon easily in surface case. Note that

$$N(u, v) = \frac{S_u(u, v) \times S_v(u, v)}{||S_u(u, v) \times S_v(u, v)||} \tag{1.3}$$

is well-defined for a regular surface $S(u, v)$ as the u and v-partial derivatives

$S_u(u,v)$ and $S_v(u,v)$ are non-parallel, and the term $S_u(u,v) \times S_v(u,v)$ never vanishes. Even if the surface $S(u,v)$ is regular, the offset surface $O(u,v)$ may have certain areas where the partial derivatives $O_u(u,v)$ and $O_v(u,v)$ become almost parallel. This problem occurs generically around the area where the surface $S(u,v)$ has curvature close to $1/d$ and the offset is taken to the concave side of the surface (see Figure 1.2).

More serious problems occur when offsetting is used in practice because not all of the regions in offset curves and surfaces are desired in CAD/CAM applications. Let us revisit the NC machining application where the 2D cutter tool path is computed from offsetting the boundary of the designed surface. As shown in Figure 1.1 (c), when the cutter tip passes the region where the surface is more concave than the inverse of its curvature, the tool should not follow all of the surface boundaries as it is, but change the direction abruptly. Otherwise, the tool is going to over-mill the surface of the object and penetrate the part surface. Hence, instead of defining the literal offset curves and surfaces using Equation (1.1) and (1.2), research focuses have been on detection of the potential self-intersections in offset curves and surfaces and trimming the redundant region from offset curves and surfaces.

Unfortunately, identifying the self-intersections and trimming the redundancies from offset curves and surfaces has long been considered as one of the most challenging problems in geometric processing [6]. Because of the aforementioned reasons, it is not easy to propose a stable algorithm from trimming offset curves and surfaces properly. Also, the near-singular areas of offsets introduce serious numerical instability, in particular, when computing the self-intersection curves of $O(u,v)$ and consequently trimming all redundant parts of $O(u,v)$ that are closer than the distance d to some other parts

of the progenitor surfaces.

These difficulties have promoted active research publications in computer-aided geometric design, including survey articles on offset curve/surface approximation techniques [18, 63, 72] and even a book on PH curves/surfaces [25]. Compared with these research problems, however, the previous work on computing the self-intersection of offset curves/surfaces and trimming offset redundancies has been quite limited.

In this thesis, we tackle the problem of identifying the self-intersections of offset curves and surfaces even when offsets contain degenerate singularities. We also propose a method to trim the redundant regions from offset curves and surfaces using the detected self-intersections. *Redundancy* throughout this thesis is defined as the region where the shortest distance from the point on the offset to the progenitor curve and surface is smaller than the offset distance. In the case of the ball cutter path, the path will cover the original surface without overcutting if we exclude those redundant regions from the ball cutter path. The scope of this thesis is on handling the offset curves and surfaces of parametrized curves and surfaces with the rational polynomial presentation. In our experiment, the input progenitor curves and surfaces will be given as Bézier and B-spline curves and surfaces. However, the methodology itself can be expanded easily to other general algebraic curves and surfaces.

1.2 Research Objectives and Approach

In geometry processing, the intersection between two parametric curves is a set of discrete points, and the intersection between two parametric surfaces is a set of intersecting curves. These intersections are also realized as the solution

of the following implicit algebraic functions.

$$C_1(t) - C_2(r) = 0 \tag{1.4}$$

$$S_1(u, v) - S_2(s, t) = 0 \tag{1.5}$$

The intersection points or curves are then found by finding the zero set of the implicit function Equation (1.4) or (1.5). The solution is expressed as a set of the points in tr-space in case of curves, or implicit algebraic curves in $uvst$-space in case of surfaces. The solution of Equation (1.5) is again the solution of three equations in u, v, s, t: $f_1(u, v) - f_2(s, t) = 0$, where $f = x, y, z$ coordinates of the two surfaces S_1 and S_2. When $S_1(u, v)$ and $S_2(s, t)$ are rational, the intersection curve is often a non-rational algebraic space curve of relatively high degree in general. Kim and Elber [21] propose a method to find the zero set of the implicit function of a high degree with geometric constraints and approximate the solution curve with low degree curve segments.

The solution of self-intersections is formulated in a similar manner. Instead of two different curves(surfaces), the implicit equations are formulated by copying the same curve(surface), but with different parametrizations as follows.

$$C(t) - C(r) = 0 \tag{1.6}$$

$$f(u, v) - f(s, t) = 0 \text{ for } f = x, y, z \tag{1.7}$$

The problem in this formulation is how to deal with the trivial solution: $t = r$ or $(u, v) = (s, t)$, which is the result of a curve(surface) always completely overlapping with itself. For the rational surfaces, there is a systematic way of removing the trivial solution from the surface self-intersections [70]. In case of offset curves and surfaces, however, non-rational terms in the formulation

hinder removing trivial solutions from the solution of self-intersections. To handle the non-rationality in offset curves and surfaces, it is required to develop a different way of formulating the constraint equations in the parameter space to detect self-intersection while eliminating the trivial and other redundant solutions from the solutions of the offset curve/surface self-intersections.

In this thesis, we introduce a geometric framework to detect the self-intersections of offset curves and surfaces and trim the redundant regions. We figure out complex topological behaviors resulted from offset self-intersections not in the Euclidean space, but the parameter domain of the curves and surfaces. However, geometric constraints observed in the Euclidean space are also integrated into designing an algorithm for trimming self-intersections.

We first focus on identifying the topological structure of trimmed offset curves and surfaces. We formulate a set of implicit constraint equations, the solution of which yields the self-intersections of offset curves and surfaces. The solution of the equations is computed using the existing multivariate geometric equation solver, such as IRIT system [41]. The equations fed to the IRIT system are constructed based on the geometric constraints from an isosceles relation between two points of the progenitor curves/surfaces and their corresponding offset points that intersect each other. Several inequality constraints help to trim redundancies from the solution of self-intersection equations. For offset curves, two equations in the tr-space are formulated to find self-intersection points, while eliminating the trivial solution such that $t = r$. In the case of offset surfaces, three equations in the $uvst$-space are formulated to detect self-intersections and the trivial solution for $(u, v) = (s, t)$ and other redundant solution are eliminated.

In implementing offset trimming, we have observed that the existing multi-

variate equation solver becomes numerically unstable and even fails to produce the solution when tracing the solution near the singular regions of offset surfaces. Computations in these regions become so unstable as the derivatives almost vanish. To overcome these difficulties, self-intersections around near-singular regions are handled in a separate routine: the computation of offset self-intersections is executed with a more numerically robust and stable surface structure. Potential pathological behaviors observed around near-singular regions are deviated by employing the new substituted structure.

The multivariate solver we used in computing the self-intersections heavily relies on curve or surface subdivision in solving the given constraint equations: the entire domain of curves and surfaces is subdivided, and a set of constraints is tested until the solver guarantees the existence of the solution or the subdivision tolerance reaches the specified value. Subdivision of curves and surfaces is, however, an expensive process. Even though some parts of curves and surfaces are trimmed out by the inequality constraints, subdivision continues to occur because the solver must test trivial solutions in self-intersections in the remaining regions of curves and surfaces. Supposed that two points are not trivial unless the difference between them in the parametric space is smaller than ϵ, we have to subdivide the domain of the given curves and surfaces again and again until the size of subdivided curve or patch becomes below ϵ.

In the second part of this thesis, we propose a method to accelerate detecting and trimming self-intersections and redundancies of offset curves and surfaces while maintaining a similar or even higher level of robustness and accuracy. The acceleration is accomplished by using the *bounding volume hiearchy* (BVH), which encloses the actual geometry of an offset curve or surface with simpler geometric objects, thus reducing the computational cost of geometric

tests involved in offset self-intersection detection. Whereas the BVH of rational curves or surfaces is often constructed from positions of the curves or surfaces, the proposed BVH is constructed from both bounding volumes and bounding normals of the progenitor curves or surfaces. The bounding volumes of offsets are usually thicker than those of the progenitors because of scaling the normals with the offset radius.

Finally, we relate the problem of trimming offset curves and surfaces to another challenging problem in CAD/CAM: finding the *Voronoi diagram* of freeform surfaces. From the trimmed offset surfaces with varying offset radius, we derive the Voronoi diagram of a freeform surface in 3D, separating the 3D space into a set of cells, the boundaries of which are loci of points that are equidistant to at least two different points on the given surface.

1.3 Contributions and Thesis Organization

The main contribution of this work can be summarized as follows, based on a few unique aspects of the proposed approach:

- We raise an important research issue for the offset surface trimming problem. The near-singularity of an offset surface is a generic nature that can be observed in the offset surface evolution. Whenever an offset is taken to the concave side of a freeform surface, the singularity starts to develop naturally for almost every offset radius. A systematic way of handling this generic problem is very critical for practical applications of offset surface trimming in the computer-aided geometric design.

- As an important first step towards handling the near-singular offset surfaces in a computationally stable way, we introduce a unique way of

formulating the constraints on offset self-intersections in an offset-free manner, i.e., using the input surface $S(u, v)$ and its partial derivatives only. Chapter 5 has more detailed technical discussions on this issue.

- We focus attention on revealing the branching structure of the offset self-intersection trimming curves on the offset surfaces. The branching structure is evident in the zoom-in views of many test examples of this thesis. This important feature has been overlooked in previous work. The main computational difficulty is in detecting the terminal endpoint of each branch, where the offset surface has a tangential self-intersection. We show that the bivariate representation of freeform surfaces provides an effective tool for attacking this non-trivial problem. The uv-solution curve in the parameter domain gives a stable way of checking whether the curve tracing has reached at the tip of a branch.

- The correct topology of the branching structure is determined by identifying all the junctions where multiple branches meet and are correctly connected. Based on the loop construction for the redundant trimming region, we present an approach to stitching pairs of matching curve segments on the loop.

- The process of trimming offset surface is further accelerated with the introduction of the new bounding volume hierarchy and the subsequent trimming constraints. This acceleration enables us to produce trimmed offset surfaces more efficiently than the subdivision-based constraint equation solving techniques, which makes the trimming algorithm more practical.

- Based on the efficient offset trimming, we have explored the Voronoi diagram construction for freeform surfaces. The construction of the 3D Voronoi structures of freeform surfaces has been tried yet, to our best knowledge.

The rest of this thesis is organized as follows. Chapter 2 summarizes some fundamental theories on curves and surfaces that are also the essential building blocks in understanding this thesis. In Chapter 3, we review previous work on offset curves and surfaces. We then present our scheme in the curve case in Chapter 4 where we introduce a geometric configuration that governs the self-intersection of an offset curve and propose a method to trim self-intersections and redundant regions from the self-intersection solution. The scheme is then extended to the surface case in Chapter 5, where we formulate the constraint equations to find the partial solutions of self-intersections without trivial solutions and further trim the partial solutions. The performance of this method is analyzed in Chapter 6. Based on the analysis, we propose a new approach to accelerate self-intersection detection and trimming of offset curves and surfaces. In Chapter 7, we derive the Voronoi cells of freeform surfaces in 3D from the trimmed offset surface results. Finally, we conclude this thesis with future work in Chapter 8.

Chapter 2

Preliminaries

2.1 Curve and Surface Representation

In this thesis, we represent the progenitor curves or surfaces by rational polynomial functions. Widely used rational polynomial representations on curves and surfaces are *Bézier* representstions and *B-spline* representations. Therefore, we briefly review both representations on curves and surfaces and summarize the algebraic and geometric properties of those curves and surfaces in this chapter. More details on these representations can be found in CAGD textbooks (e.g., Farin [23]).

2.1.1 Bézier Representation

A *Bernstein polynomial* is a polynomial function defined as follows.

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i \tag{2.1}$$

A Bézier curve of degree n is a curve parametrized in $t \in [0, 1]$ and expressed as a weighted sum of Bernstein basis functions as follows.

$$c(t) = \sum_{i=0}^{n} \mathbf{b_i} B_i^n(t) \tag{2.2}$$

where $\mathbf{b_i}$'s are called control points of the curve and determine the shape of the curve. Bézier curves are robust in numerical computations because of the properties listed below.

1. Convex hull property: Because Bernstein bases are always nonnegative and summed up to one, every point on a Bézier curve is included in the convex hull of control points of the curve.

2. Affine invariance: When a Bézier curve transforms under an affine transformation, control points of the transformed curve are the control points of the original curve transformed by the same transformation.

3. Endpoint interpolation: A starting and an ending points of a Bézier curve are control points of the curve, that is, $\mathbf{b_0} = c(0)$ and $\mathbf{b_1} = c(1)$.

The k-th order derivative of a Bézier curve is defined as follows.

$$\frac{\mathrm{d}^k c(t)}{\mathrm{d}t^k} = \frac{n!}{(n-k)!} \sum_{i=0}^{n-k} \Delta^k \mathbf{b_i} B_i^{n-k}(t) \tag{2.3}$$

where Δ^k is *the iterated forward difference operator* defined as $\Delta^k \mathbf{b_i} = \Delta^{k-1} \mathbf{b_{i+1}} - \Delta^{k-1} \mathbf{b_i}$. The k-th order derivative of a Bézier curve is another Bézier curve with k lower degrees, and the control points are directly computed from differencing the control points of the $(k-1)$-th order Bézier curve. When the original Bézier curve is cubic that has $\mathbf{b_0}$, $\mathbf{b_1}$, $\mathbf{b_2}$ and $\mathbf{b_3}$ as control points, for instance, the first order derivative is a quadratic Bézier curve controlled

by $3(\mathbf{b_1} - \mathbf{b_0})$, $3(\mathbf{b_2} - \mathbf{b_1})$ and $3(\mathbf{b_3} - \mathbf{b_2})$, and the second order derivative is a linear Bézier curve controlled by $6(\mathbf{b_2} - 2\mathbf{b_1} - \mathbf{b_0})$ and $6(\mathbf{b_3} - 2\mathbf{b_2} - \mathbf{b_1})$, and so forth.

A Bézier surface patch of degree $m \times n$ is a surface patch parametrized by u and v in $[0,1] \times [0,1]$ as follows.

$$S(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b_{i,j}} B_i^m(u) B_j^n(v) \tag{2.4}$$

where $\mathbf{b_{i,j}}$'s are control points of the surface and $B_i^m(u)$ and $B_j^n(v)$ are Bernstein bases of degree m and n, respectively. Bézier surface is one of a tensor product surface that is written in the matrix form of $S(u,v) = M^T(u)\mathbf{B}N(v)$. Just as Bézier curves, Bézier surface also has convex hull property, affine invariance, and endpoint interpolation.

For a Bézier surface, we can compute the partial derivatives similar to those of a Bézier curve. Here we only list the first and the second order partials of Bézier surface (in Equation (2.5), (2.6) and Equation (2.7), (2.8), (2.9)) but the higher order partial derivatives can be computed in a similar way.

$$S_u(u,v) = m \sum_{i=0}^{m-1} \sum_{j=0}^{n} (\mathbf{b_{i+1,j}} - \mathbf{b_{i,j}}) B_i^{m-1}(u) B_j^n(v) \tag{2.5}$$

$$S_v(u,v) = n \sum_{i=0}^{m} \sum_{j=0}^{n-1} (\mathbf{b_{i,j+1}} - \mathbf{b_{i,j}}) B_i^m(u) B_j^{n-1}(v) \tag{2.6}$$

$$S_{uu}(u,v) = m(m-1)\sum_{i=0}^{m-2}\sum_{j=0}^{n}(\mathbf{b_{i+2,j}} - 2\mathbf{b_{i+1,j}} + \mathbf{b_{i,j}})B_i^{m-2}(u)B_j^n(v) \quad (2.7)$$

$$S_{vv}(u,v) = n(n-1)\sum_{i=0}^{m}\sum_{j=0}^{n-2}(\mathbf{b_{i,j+2}} - 2\mathbf{b_{i,j+1}} + \mathbf{b_{i,j}})B_i^m(u)B_j^{n-2}(v) \quad (2.8)$$

$$S_{uv}(u,v) = mn\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(\mathbf{b_{i+1,j+1}} - \mathbf{b_{i+1,j}} - \mathbf{b_{i,j+1}} + \mathbf{b_{i,j}})B_i^{m-1}(u)B_j^{n-1}(v) \quad (2.9)$$

2.1.2 B-spline Representation

Compared to Bézier representation, B-spline representation provides a more general form to express rational polynomial curves and surface. Instead of using Bernstein polynomials as bases, B-spline curves and surfaces use *piecewise polynomial* bases that give more flexible and local control than Bézier representations.

A B-spline curve is defined as follows.

$$c(u) = \mathbf{d_0}N_0^n(u) + \mathbf{d_1}N_1^n(u) + \cdots + \mathbf{d_{D-1}}N_{D-1}^n(u) \quad (2.10)$$

where $N_i^n(u)$'s are B-spline bases of degree n and $\mathbf{d_i}$'s are *de Boor points* or control points of the B-spline curve. Piecewise control comes from splitting the interval of the curve based on a *knot sequence* and evaluating the curve with linear interpolations of control points on each interval separately. The multiplicity of knots determines the continuity of the curve, and the number of distinct knots determines the number of the interval split. The number of control points D and the number of knots K in a B-spline curve are related, satisfying $D = K - n + 1$.

A basis function $N_i^n(u)$ of a B-spline curve is recursively defined as follows:

$$N_i^0(u) = \begin{cases} 1 & \text{if } u_{i-1} \le u \le u_i \\ 0 & \text{else} \end{cases} \tag{2.11}$$

$$N_i^n(u) = \frac{u - u_{i-1}}{u_{i+n-1} - u_{i-1}} N_i^{n-1}(u) + \frac{u_{i+n} - u}{u_{i+n} - u_i} N_{i+1}^{n-1}(u) \text{ for } n > 0 \tag{2.12}$$

A B-spline curve is also evaluated by repeating linear interpolations on control points with *de Boor* algorithm written below.

$$c(u) = \mathbf{d}_{I+1}^n(u) \quad \text{for } u_I \le u \le u_{I+1} \tag{2.13}$$

$$\mathbf{d}_i^k(u) = \frac{u_{i+n-k} - u}{u_{i+n-k} - u_{i-1}} \mathbf{d}_{i-1}^{k-1}(u) + \frac{u - u_{i-1}}{u_{i+n-k} - u_{i-1}} \mathbf{d}_i^{k-1}(u) \tag{2.14}$$

where $k = 1, \cdots, n$ and $i = I - n + k + 1, \cdots, I + 1$. B-spline curves also have affine invariance and end point interpolation properties just as Bézier curves.

A B-spline surface is also a tensor product surface defined as follows.

$$S(u, v) = \sum_{i=0}^{D-1} \sum_{j=0}^{E-1} \mathbf{d_{i,j}} N_i^m(u) N_j^n(v) = M^T \mathbf{D} N \tag{2.15}$$

where $\mathbf{D}$ is a control net of size $D \times E$ and m and n are the degrees of B-spline bases along u and v directions. What we have defined and discussed on B-spline curves are easily extended to B-spline surfaces as well.

Finally, curves and surfaces can also be represented by rational B-splines as follows.

$$c(u) = \frac{w_0 \mathbf{d_0} N_0^n(u) + w_1 \mathbf{d_1} N_1^n(u) + \cdots + w_{D-1} \mathbf{d_{D-1}} N_{D-1}^n(u)}{w_0 N_0^n(u) + w_1 N_1^n(u) + \cdots + w_{D-1} N_{D-1}^n(u)} \tag{2.16}$$

$$S(u, v) = \frac{M^T \mathbf{D_w} N}{M^T W N} \tag{2.17}$$

where w_i's are the weights of B-spline bases. $\mathbf{D_w}$ has the elements $w_{i,j}\mathbf{d_{i,j}}$ and W has the elements $w_{i,j}$. Curves and surfaces that have different weights are

called *Non-Uniform Rational B-Spline curves/surfaces* (NURBS) and have extensive use in many CAD/CAM systems.

2.2 Differential Geometry of Curves and Surfaces

Fundamentals of geometric decisions and operations in offset trimming stem from many concepts in differential geometry of curves and surfaces. In this section, we review differential geometric properties and terminology of curves and surfaces used throughout this thesis. For further details, readers refer to textbooks of differential geometry (e.g., DoCarmo [10], Spivak [80]).

2.2.1 Differential Geometry of Curves

In the differential geometric perspective, a parametrized differential curve is a mapping $f : I \to \mathbb{R}^2$ (in case of a planar curve) or $f : I \to \mathbb{R}^3$ (in case of a spatial curve), defined in an open interval $I = (a, b) \in \mathbb{R}$ such that

$$f(t) = (x(t), y(t)) \quad \text{(a planar curve)} \tag{2.18}$$

$$f(t) = (x(t), y(t), z(t)) \quad \text{(a spatial curve)} \tag{2.19}$$

where $x(t)$, $y(t)$ and $z(t)$ are differentiable. A tangent $f'(t)$ of the curve is then expressed as $f'(t) = (x'(t), y'(t))$ or $f'(t) = (x'(t), y'(t), z'(t))$. A curve $f(t)$ is said to be *regular* if $f'(t) \neq 0$ for all $t \in I$. If there exist some points on the curve such that $f'(t) = 0$, we call those points *singular points* of the curve. Many terms in differential geometry are often only defined for regular curves. The regularity of a curve is the underlying assumption for many concepts introduced below.

A regular spatial curve $f(t) = (x(t), y(t), z(t))$ can be reparametrized with

a new parameter s such that

$$s = s(t) = \int_{t_0}^{t} |f'(t)|dt = \int_{t_0}^{t} \sqrt{(x'(t))^2 + (y'(t))^2 + (z'(t))^2}dt \tag{2.20}$$

. This parametrization is called *arc-length parametrization* and simplifies many formulae on curves.

Supposed that a curve is parametrized by arc-length, $|f''(s)| = \kappa(s)$ is called a *curvature* of f. For $\kappa(s) \neq 0$, $f''(s) = \kappa(s)n(s)$ is well-defined and $n(s)$ is a unit normal of a curve at s. We call a point a singular point of order 1 when $f''(s) = 0$, whereas a singular point of order 0 when $f'(s) = 0$. If $f(t)$ is not parametrized by arc-length, the curvature $\kappa(t)$ is formulated as follows.

$$\kappa(t) = \frac{|f'(t) \wedge f''(t)|}{|f'(t)|^3} \tag{2.21}$$

The plane spanned by a normal $n(s)$ and a tangent $t(s)$ of a curve is called *an osculating plane*. A curvature shows the rate of how the curve is bent within the osculating plane. A unit vector $b(s) = t(s) \wedge n(s)$ is called a *binormal vector* at s. Then the *torsion* $\tau(s)$ of a curve at s is defined as $b'(s) = \tau(s)n(s)$. A torsion represents the rate of how the curve deviates from the osculating plane.

$t(s)$, $n(s)$ and $b(s)$ are related to each other as shown in Equations (2.22), forming a local frame of a curve at s, which is called a *frenet frame*.

$$\begin{aligned} t(s)' &= \kappa(s)n(s) \\ n(s)' &= -\kappa(s)t(s) - \tau(s)b(s) \\ b(s)' &= \tau(s)n(s) \end{aligned} \tag{2.22}$$

In case of a planar curve $f(t) = (x(t), y(t))$, a normal vector $n(t)$ and a

curvature $\kappa(t)$ are simplified as follows.

$$n(t) = \frac{(y'(t), -x'(t))}{\sqrt{x'(t)^2 + y'(t)^2}} \tag{2.23}$$

$$\kappa(t) = \frac{x'(t)y''(t) - x''(t)y'(t)}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}} \tag{2.24}$$

In this case, an osculating circle of a curve $f(t)$ at t is a circle that has $\frac{1}{|\kappa(t)|}$ as a radius and $f(t) + \frac{1}{|\kappa(t)|}n(t)$ as a center. This circle touches the curve, having the same curvature as the curve at t.

2.2.2 Differential Geometry of Surfaces

A parametrized differential surface is a mapping $S : U \to \mathbb{R}^3$ defined on U, a subset of $\mathbb{R}^2$ such that

$$S(u, v) = (x(u, v), y(u, v), z(u, v)) \tag{2.25}$$

where $x(u, v)$, $y(u, v)$ and $z(u, v)$ are differentiable. For a surface to be *regular*, this mapping must be one-to-one, or equivalently satisfy the following constraints:

$$\frac{\partial S}{\partial u} \wedge \frac{\partial S}{\partial v} \neq 0 \quad \text{for} \quad all \ (u, v) \in U \tag{2.26}$$

.

A normal map (gauss map) of $S(u, v)$ is another mapping $N : q \to \mathbb{R}^3$ such that

$$N(q) = \frac{S_u \wedge S_v}{|S_u \wedge S_v|}(q) \quad \text{for} \quad all \ q \in S(u, v) \tag{2.27}$$

.

Two relevant geometric entities explain the local geometric properties of a surface: the first and the second fundamental forms of the surface. First, the

first fundamental form of the surface $S(u, v)$ is defined as follows.

$$E =< S_u(u, v), S_u(u, v) > \tag{2.28}$$

$$F =< S_u(u, v), S_v(u, v) > \tag{2.29}$$

$$G =< S_v(u, v), S_v(u, v) > \tag{2.30}$$

where $< \cdot\ ,\ \cdot >$ is a dot product. The first fundamental form is involved in various measurements of the surface, such as the area of regions on the surface, the length of curves on the surface, and the angle between tangents on the surface.

Next, the *second fundamental form* of the surface $S(u, v)$ is defined as follows.

$$e =< N(u, v), S_{uu}(u, v) > \tag{2.31}$$

$$f =< N(u, v), S_{uv}(u, v) > \tag{2.32}$$

$$g =< N(u, v), S_{vv}(u, v) > \tag{2.33}$$

The first and second fundamental forms are closely related to definitions of different types of curvatures on the surface. Among those curvatures, widely used ones are the gaussian curvature K and the mean curvature H of $S(u, v)$, which are defined as follows.

$$K = \frac{eg - f^2}{EG - F^2} \tag{2.34}$$

$$H = \frac{eG - 2fF + gE}{2(EG - F^2)} \tag{2.35}$$

Chapter 3

Previous Work

The definition of offset curve and surface (under the name of parallel curve and surface) can be found in many textbooks of differential geometry [10, 80]. The differential properties of offset curves and surfaces are often given as exercise problems in these textbooks. In the field of CAGD, researches on offsetting curves, surfaces, and solids have started to gain attention in the 1980s while attempting to define geometric operations systematically. The long history of offset curve and surface computation since then has been well-documented in survey articles [18, 63, 72] and textbooks [25]. In his survey paper [63], Maekawa categorizes researches on offset curves and surfaces into four groups: offsetting the particular types of curves and surfaces, the approximation of offsets, self-intersections of offsets, and offset curves and surfaces with non-Euclidean or non-uniform distance metric. In this chapter, we briefly summarize the previous work on offset curve and surface computation. We review the researches that focus on offset curves and offset surfaces separately, al-

beit some of the methods apply to both curves and surfaces. Also, we discuss researches about generating offset curves on surface in the last section.

3.1 Offset Curves

Offsets of planar curves have gained many research interests from the early ages in the CAGD community. Farouki [26] gives greater details of geometric properties of offset curves and also algebraic properties of the exact offset curves [27]. As exact offset formulation involves a nonrational term derived from the unit normal, handling nonrationality has been always an important issue in offset curve and surface computation. To handle nonrationality in offset curves (and surfaces as well), some researchers have focused on the particular type of curves that simplifies the offset formulation. For instance, Farouki and his colleagues have published many papers in Pythagorean-Hodograph(PH) curves, the special case of rational curves, of which offsets are also represented in terms of rational functions. (See the survey [28] and the book [25] for further details.) Analogous to PH curves, researchers have also investigated the class of rational surfaces that generates rational offset surfaces. Krasauskas and Peternell [50] discuss the characteristics of such rational offset surfaces.

Another direction to overcoming the nonrationality of offset curves is to approximate offset curves with other rational functions. Many approximation methods employ an iterative refinement strategy in approximating offset curves for a prescribed tolerance: B-spline curves are subdivided into smaller curves when the approximation errors are beyond a tolerance value and approximated independently for the subdivided curves. Elber et al. [18] provide a comparative study of the existing offset curve approximation meth-

ods [14, 15, 17, 37, 38, 49, 56, 71, 83] both quantitively and qualitatively. Here, they compare the quality of the approximated curves based on the number of control points in the approximated curves.

The mathematical definition of an offset curve generates a curve that consists of loci having a constant distance from the progenitor curve, but the definition itself does not guarantee that the minimum distance between the progenitor and the offset curves is also equal to the offset radius for the entire domain of the curve. In other words, an offset curve may contain some parts that are globally closer to the progenitor curve than the offset radius. Also, if a point on the progenitor curve has the curvature equal to the inverse of the offset radius, the corresponding point on the offset curve becomes a cusp in which the tangent is not well-defined. There has been an extensive number of researches that tackle the problems of identifying potentially pathological behaviors in an offset curve such as discontinuities, cusps, and self-intersection points, and computing the topological and geometrical structure of an offset curve. Maekawa et al. [61] find the singular points such as cusps and self-intersection points in planar offset curves by solving the bivariate constraint equations of offset curves. Solutions of the constraint equations are computed numerically based on interval arithmetic. Elber [22] also trims planar offset curves by removing redundant parts in offset curves caused by singularities. Based on the observation that parts of an offset curve that are closer to the progenitor curve than the offset radius must be eliminated, he formulates the bivariate constraint equations such that the distance between the offset curve and the progenitor curve must be not smaller than the offset distance.

Another widely used technique to compute the topology of an offset curve is to decompose and analyze offset curve equations with a subresultant sequence.

Caravantes et al. [9] have proposed to detect significant points by solving several univariate polynomials through the subresultant space and topologically sort the solutions of these equations to yield the offset curve. However, their method is somewhat slow because the equations to solve are complex, and there are no guarantees of applying the method to general rational curve functions.

Computing the topology of an offset curve by finding discontinuities and self-intersections in the exact offset curve and trimming them away is based on solving complex constraint equations of relatively high degree, which deteriorates the performance of the method. Therefore, different approaches have been proposed: an offset curve is first approximated, and unnecessary parts are trimmed from the approximated offset curve. Kim et al. [46] demonstrate the robustness in offset curve trimming by approximating rational curves in biarcs. Arc-based approximation provides simple computation of the curvature of curves, therefore making it easy to identify cusps from the offset curves. They adapt the point projection method of Hu and Wallner [39] using osculating circles, and find a robust and stable self-intersection detection technique on planar offset curves. Lee et al. [57] further enhance the robustness of the approach and accelerate the algorithm by building a bounding volume hierarchy of bounding circular arcs (BCAs) to compute self-intersections in planar offset curves. Although offset trimming of a planar curve is now considered almost a solved problem thanks to these researches, the approximation nature causes unguaranteed behaviors in the offset curve. The parametrization of the approximated curve is different from the original curve, and the topological decisions are made in the new arc-parameter domain instead of the original curve domain, which only gives the empirical similarness of the result.

3.2 Offset Surfaces

Farouki [24] seeks geometric properties of offset surfaces in detail with implementation results on offset surfaces. However, this work only handles the normal behaving offset surfaces and excludes the discussion of the pathological behaviors of offset surfaces. Compared with the robust results presented in offset curve trimming, the complexity and non-trivial situations of offset surface trimming hinder active researches in offset surface trimming. To circumvent these difficulties, the majority of the previous approaches in offset surface trimming are either on the approximation of non-rational offsets using rational surfaces or on the special cases where the offset surfaces are represented in terms of rational surfaces.

There are very few previous results for offset surface trimming [3, 6, 62, 78, 88]. The majority of offset trimming results are on triangular meshes, where the exact offset surfaces are approximated with meshes. The mesh offset algorithms are mainly based on grid structures [69, 87], and consequently approximate the trimmed offset solutions in relatively low grid resolutions. There are a few exceptions where the offset trimming can be carried out directly on the offset mesh [8, 12, 53]. For the problem we consider in the current work, the near-singular regions generate a large number of long and thin surface elements, the intersection of which causes computational difficulties in the determination of correct topology among these spike-like features clustered together. Consequently, the grid or mesh-based approaches have limitations in handling these thin features. In particular, Campen and Kobbelt [8] and Kyung et al. [53] compute the offset mesh as the Minkowski sum of an input mesh and a sphere, where the offset sphere is approximated by a polygonal

mesh. Their trimming algorithms are highly efficient and robust; however, the trimming results are limited to the accuracy of sphere approximation.

As mentioned above, the majority of conventional methods deal with the offset surface trimming problem using rational surface approximations to the exact but non-rational offset surface [78]. Thus the offset self-intersection can be reduced to a slightly easier problem of intersecting rational surfaces. Nevertheless, around the near-singular regions of an offset surface, it is extremely challenging to approximate the offset surface with rational surfaces that can faithfully preserve the intersection topology or guarantee the approximation error. This limitation has promoted different approaches such as Maekawa et al. [62] that can directly compute the offset surface self-intersection from the given surface definition:

$$S(u,v) + d \cdot N(u,v) = S(s,t) + d \cdot N(s,t), \tag{3.1}$$

for $(u,v) \neq (s,t)$. Nevertheless, the numerical instability is again generic even in this case, for the problem of self-intersecting a bivariate surface around near-singular regions of the surface. The curve tracing of Maekawa et al. [62] is based on the Runge-Kutta iteration of Aomura and Uehara [3] to a differential equation derived from this equation. Wang [88] proposed a different formulation of differential equations and dealt with discrete singular self-intersections only. Nevertheless, the conventional methods have no explicit discussions on the branching structure of the offset trimming curves.

Regardless of using either approximate or exact offset representations, the offset self-intersection and trimming methods convert the given problem to a curve arrangement problem in the uv-parameter space of the given input surface. The approximation-based methods usually have a less serious problem in

constructing the arrangement of curve segments, mainly due to the simplification of the given problem using low degree surfaces for approximation. On the other hand, for direct methods based on the exact offset formula, there is a higher chance of getting into difficulty deciding the correct arrangement in the uv-parameter domain, because of the collapsing nature of the offset around near-singular regions. In this respect, the near-singularity issue we raise in this thesis is important for the offset surface trimming problem.

Mizrahi et al. [65] demonstrated the offset surface trimming as a special case of the Minkowski sum computation for freeform surfaces. In some sense, the previous results on the Minkowski sum computation (see those referenced in Mizrahi et al. [65]) can be used for offset surface trimming up to a certain level. But their general approach has limitations in handling the near-singular surface features we consider in this thesis.

3.3 Offset Curves on Surfaces

The researches introduced in the previous sections investigate offset curves and surfaces that have a constant offset distance from the progenitors in Euclidean space. Some CAD/CAM applications, however, necessitate offsets with non-Euclidean distance metric. A geodesic offset curve, for instance, is a curve of which locus has a constant distance d to the progenitor curve lying on a surface. Different from a Euclidean offset curve, the offset distance of the geodesic offset curve is measured along the geodesic path from the progenitor curve so that the geodesic offset curve also lies on the same surface. In CAD/CAM applications, geodesic offsets are used in the construction of linkage curves between two blending surfaces [33], tool path generation of 3-axis ball-cutter

milling with constant scallop height [30], planning spray gun trajectories [4], or automatic fiber placement [75].

In the field of CAGD, Patrikalakis and Bardis [66] have first proposed a computational approach to compute a geodesic offset curve on a NURBS surface. They derive the differential equations to compute the geodesic direction of a curve on a surface, which is perpendicular to the tangent of the curve. Then they compute the sample points in uv-parametric space on the geodesic path using a numerical integration method such as Runge-Kutta iteration. The computed points are interpolated to construct the spline curve that approximates the exact geodesic offset curve. Wolter and Tuohy [89] enlist a geodesic offset curve as an example of applications for generating a procedurally defined high-ordered curve. As the closed-form equation of the geodesic offset curve is complicated or even not feasible to formulate, they also derive the differential equations of the curve and solve the equations using a numerical method. These conventional approaches are also adopted by other researchers [7, 31].

As solving differential equations using Runge-Kutta-like numerical methods is quite complicated and time-consuming, there have been alternative approaches to approximate geodesic offset curves. Ulmet [85] provides more industry-friendly and practical methods by approximating geodesic lines with linear vectors lying on the tangent plane of the curve on the surface, taken from the direction perpendicular to the tangent of the curve. They also suggest another alternative by projecting the tangent vector back onto the surface. Those methods pursue the efficiency of computation while sacrificing the quality of the approximation. They also emphasize the importance of reparametrization in computing geodesic paths across multiple patches with different parametrization. Geodesic paths are also actively researched in the

context of generating constant scallop height in 3-axis milling with ball cutters [30, 76, 81, 44]. By planning paths of the cutters as following the geodesic offset lines, the amount of redundancy in machining can be minimized. Researches sofarmentioned handle the geodesic offset curves on freeform surfaces. There has also been an extensive number of researches about computing geodesic offsets on triangular meshes which we will not review in detail in this thesis [36, 47, 59, 90].

Chapter 4

Trimming Offset Curve Self-intersections

In this chapter, we introduce a framework to detect and trim self-intersections of an offset curve when the progenitor curve is the planar curve defined in R^2. The basic idea of the proposed method is to detect the points on the offset curve where self-intersection occurs by solving constraint equations formulated in parametric tr-space. This framework applies similarly to find self-intersections of an offset surface, despite that the dimension of parameters increases from t to (u, v).

Detected self-intersection points subdivide the offset curve into several curve segments. Redundant segments are eliminated to construct the final trimmed offset curve, which does not include any intervals that are closer than the offset radius of d. Trimming self-intersections from an offset curve is a much simpler process than that of an offset surface. Still, it is worthwhile to analyze the trimming process because several concepts proposed here can

be shared or expanded to design an algorithm for trimming an offset surface that we will mention from Chapter 5. In this chapter, we grasp the intuition of the geometric relation on the self-intersection points of offsets via offset curves in R^2. We also propose a method to trim the redundant intervals from offset curves. Several examples of trimmed offset curves are also presented to validate the proposed method.

We first identify constraint equations governing self-intersection points of an offset curve. For a parametric curve $C(t) = (x(t), y(t))$ in R^2, an offset curve $O(t)$ is defined as follows:

$$O(t) = C(t) + d \cdot N(t) \tag{4.1}$$

where d is an offset distance and $N(t)$ is a unit normal field of $C(t)$, that is,

$$N(t) = \frac{(y'(t), -x'(t))}{\sqrt{((x')^2 + (y')^2)}} \quad \left(\; for \; \frac{dC(t)}{dt} = (x'(t), y'(t)) \; \right) \tag{4.2}$$

. If two arbitrary points $C(t)$ and $C(r)$ $(t \neq r)$ on the curve meet each other when offset, $C(t)$, $C(r)$ and their common intersection point $O(t) = O(r)$ have to form an equilateral triangle. Figure 4.1 illustrates the geometric configuration between two points $C(t)$, $C(r)$ and their corresponding offset point $O(t)$ (or $O(r)$). Based on this configuration, we formulate two constraint equations that must be satisfied at every self-intersection point on the offset curve.

As shown in Figure 4.1,

$$\cos\alpha = \frac{h}{d} = \frac{\|C(r) - C(t)\|}{2d}, \tag{4.3}$$

and equivalently

$$\frac{\langle N(t), C(r) - C(t) \rangle}{\|C(r) - C(t)\|} = \frac{\|C(r) - C(t)\|}{2d} \tag{4.4}$$

$$\frac{\langle N(r), C(t) - C(r) \rangle}{\|C(t) - C(r)\|} = \frac{\|C(t) - C(r)\|}{2d}. \tag{4.5}$$

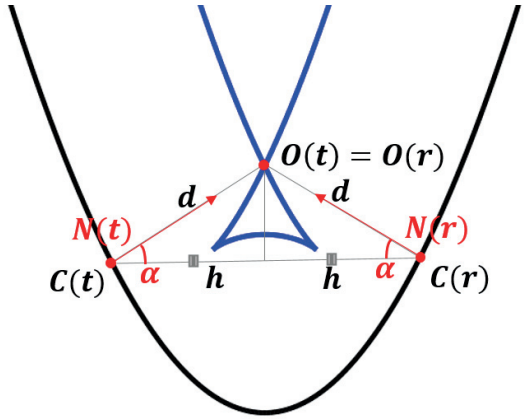


Figure 4.1: The eqilateral triangle relation at offset curve self-intersection.

Non-rational terms in Equation (4.4) and (4.5) can be converted into rational terms by squaring both sides of the equations as follows:

$$4d^2 \langle M(t), C(r) - C(t) \rangle^2 = \|C(r) - C(t)\|^4 \|M(t)\|^2 \tag{4.6}$$

$$4d^2 \langle M(r), C(t) - C(r) \rangle^2 = \|C(t) - C(r)\|^4 \|M(r)\|^2 \tag{4.7}$$

where $N(t) = M(t)/\|M(t)\|$ and $N(r) = M(r)/\|M(r)\|$. Instead of solving Equation (4.4) and (4.5), we solve Equation (4.6) and (4.7) that have twice larger degrees but are only composed of rational terms.

Equation (4.6) and (4.7) are necessary conditions for the self-intersection points of the offset curve; every self-intersection point is a solution of Equation (4.6) and (4.7), but every point such that $t = r$ also satisfies the above equations. To eliminate the trivial solutions such that $t = r$, we introduce an inequality constraint (4.8) that filters out the solutions in which t and r are too close.

$$(t - r)^2 > \epsilon \tag{4.8}$$

Offset curves can be subdivided at the self-intersection points so that important topological decisions are made on these points; the offset curve segments bounded by the self-intersection points or the boundary points are eliminated or kept based on these decisions. To decide whether a curve segment be eliminated or not, we pick a sample point inside each segment and compute the minimum distance from this point to the progenitor curve $C(t)$. Any point can be selected, but we pick the middle point of the curve where the parameter of the point is a center of the domain of the curve. If the minimum distance is smaller than the offset distance d, there exists another point $C(t_1)$ on the curve that is closer than the offset distance, and the selected point belongs to the curve segment that penetrates to other curve segments or itself. Because the whole curve segment behaves in the same topological way, the entire curve segment, including the point can be eliminated as well. Finally, trimmed curve segments are merged to construct a single closed curve, a single open curve, or a combination of closed and open curve segments.

4.1 Experimental Results

The proposed method is validated with various offset curves in R^2. In Figure 4.2, the progenitor curves are presented in black lines, and trimmed offset curves are shown in various colors. For each progenitor curve, we generate offset curves by both increasing and decreasing offset distances. In Figure 4.2 (a), self-intersection begins to appear when the original curve is offset into the concave side. When the curve is offset into the concave side, the region where normal flips (as known as " fishtail") begins to appear. (See the blue line in Figure 4.1.) Fishtail segments appear whenever the offset distance is larger

than the radius of the curvature of the curve. Our method trims the offset curve segments containing fishtails because the projected distances of these segments to the progenitor curve are smaller than the offset distance.

In Figure 4.1 (b), topologies of the offset curves are different from that of the original curve depending on the offset distance; the original curve is a single open curve, whereas the offset curves start to show islands inside when the offset distance increases. This topological difference of the offset curves is also observed in Figure 4.1 (d) as well. Here, one single closed progenitor curve is offset to several disconnected closed offset curve segments.

Though we do not find any discrepancies in the current trimming result, the choice of ϵ in Equation (4.8) may cause inaccurate trimming results. If there indeed exist local self-intersections within ϵ in $O(t)$, we do not have a method to detect those self-intersections, yet. Especially, this situation happens when self-intersections occur in the region where the offset distance is just larger than the curvature of radius.

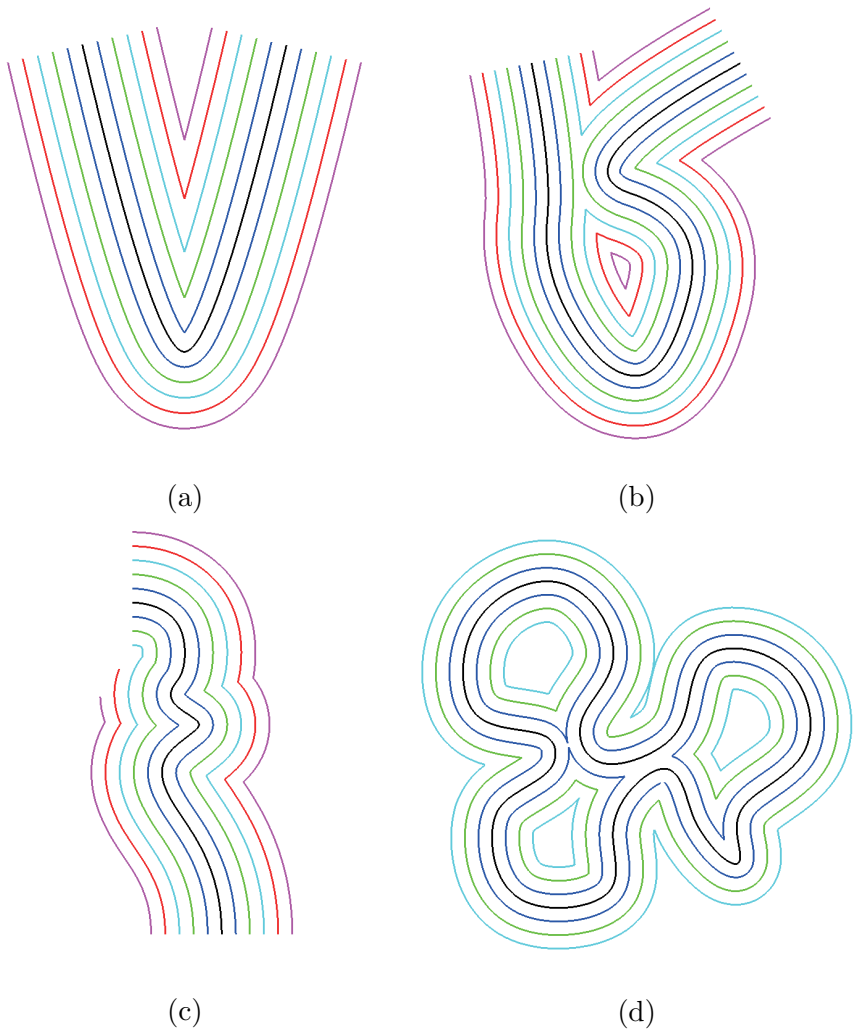


Figure 4.2: Trimmed offset curves. The progenitor curves are shown in black.

Chapter 5

Trimming Offset Surface Self-intersections

5.1 Constraint Equations for Offset Self-Intersections

Constraint equations for the self-intersections of an offset surface are constructed in a similar way to those of an offset curve. We first consider the derivation of three polynomial equations, the common solution of which produces a superset of the following offset self-intersection curve in the (u, v)-domain:

$$\mathcal{S} = \{(u, v) \mid O(u, v) = O(s, t), \text{ for some } (s, t) \neq (u, v)\}. \tag{5.1}$$

Figure 5.1 shows a generic configuration for the self-intersections of an offset surface, where an offset $O(u, v)$ from $S(u, v)$ meets another offset $O(s, t)$ from $S(s, t)$. Then the three points $S(u, v)$, $S(s, t)$, and $O(u, v) = O(s, t)$, form an isosceles triangle. Using geometric constraints derived from this triangle,

we formulate three polynomial equations in (u, v, s, t). The solutions to these equations include redundant solutions. We discuss how to eliminate redundant solutions using other constraints formulated as inequalities.

Remark: One very significant and unique feature in our derivation of these constraint formulae is that the offset surface $O(u, v)$ or $O(s, t)$ never appears explicitly in the final form of equality and inequality constraints. Dependent only on the input surface $S(u, v)$ and the first and second partial derivatives of $S(u, v)$, the constraint solving process should be as irrelevant as possible from the near-singularity of the offset surface in some areas.

5.1.1 Coplanarity Constraint

In the triangle of Figure 5.1, the three vectors $N(u, v)$, $N(s, t)$, and $S(u, v) - S(s, t)$ are coplanar. A necessary condition for the offset self-intersection at $O(u, v) = O(s, t)$ can be formulated as follows:

$$\det\left[S_u \times S_v, S_s \times S_t, S(u, v) - S(s, t)\right] = 0, \tag{5.2}$$

where $S_u \times S_v = S_u(u, v) \times S_v(u, v)$ and $S_s \times S_t = S_s(s, t) \times S_t(s, t)$.

When $S(u, v)$ is a bivariate polynomial surface of degree (m, n), Equation (5.2) is a four-variate polynomial equation of degree $(3m-1, 3n-1, 3m-1, 3n-1)$ in (u, v, s, t). For a bicubic surface $S(u, v)$, the polynomial has degree 8 in each variable. Compared with other equations to be derived below, this equation has a lower degree and therefore plays an important role in accelerating the equation solving for the formulated constraints.

5.1.2 Equi-angle Constraints

In Figure 5.1, let h denote the half-length of the base of the triangle: $h = \|S(u,v) - S(s,t)\|/2$, then we have

$$\cos\alpha = \frac{h}{d} = \frac{\|S(u,v) - S(s,t)\|}{2d}, \tag{5.3}$$

and equivalently

$$\frac{\langle N(u,v), S(s,t) - S(u,v)\rangle}{\|S(u,v) - S(s,t)\|} = \frac{\|S(u,v) - S(s,t)\|}{2d}. \tag{5.4}$$

Replacing $N(u,v)$ by $(S_u \times S_v)/\|S_u \times S_v\|$, we have

$$2d\ \frac{\langle S_u \times S_v, S(s,t) - S(u,v)\rangle}{\|S_u \times S_v\|} = \|S(u,v) - S(s,t)\|^2. \tag{5.5}$$

The term $\|S_u \times S_v\|$ is represented as the square-root of a function, which is difficult to handle in conventional multivariate equation solvers. Squaring the above equation, we get the following square-root-free equation:

$$4d^2 \langle S(u,v) - S(s,t), S_u \times S_v\rangle^2 = \|S(u,v) - S(s,t)\|^4 \|S_u \times S_v\|^2. \tag{5.6}$$

Similary, we can derive the symmetric equation as well:

$$4d^2 \langle S(u,v) - S(s,t), S_s \times S_t\rangle^2 = \|S(u,v) - S(s,t)\|^4 \|S_s \times S_t\|^2. \tag{5.7}$$

5.2 Removing Trivial Solutions

To avoid trivial solutions: $(u,v) = (s,t)$, we also employ the following inequality constraint:

$$(u-s)^2 + (v-t)^2 > \epsilon^2, \tag{5.8}$$

for a small constant $\epsilon > 0$.

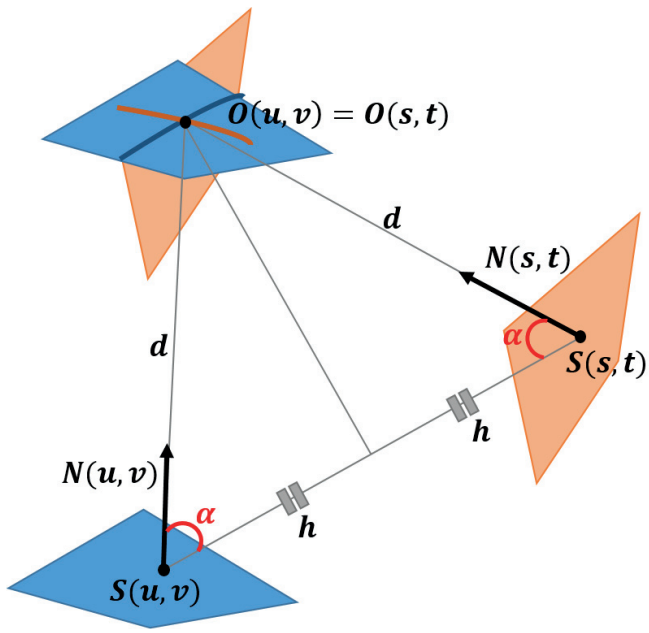


Figure 5.1: The eqilaternal triangle from an offset self-intersection condition.

Due to the symmetry of (u, v) and (s, t) in Equations (5.2)-(5.7), it is clear that duplicate solutions are generated for these constraints. We may avoid duplicate solutions as well as trivial solutions, by considering one linear inequality constraint: $u - s > \epsilon$ (or $v - t > \epsilon$), instead of the above quadratic inequality. However, for the purpose of checking the stability of computing results (as discussed in the first paragraph in Chapter 5), we keep both copies of the duplicate solutions.

5.3 Removing Normal Flips

The normal vector $O_u \times O_v$ of the offset surface $O(u, v)$ should be in the same direction as the normal vector $S_u \times S_v$ of $S(u, v)$, the condition of which can

be formulated as follows:

$$f(u,v) = \langle O_u \times O_v, S_u \times S_v \rangle > 0. \tag{5.9}$$

This inequality constraint is very useful in the elimination of redundant offset surface patches that result from the surface area where one (but not both) of the principal curvatures to the concave side is larger than $1/d$, thus introducing local self-intersections in the offset surface. The formula for $f(u,v)$ is derived in 5.A:

$$f(u,v) = \|M(u,v)\|^2 + d \cdot \frac{p(u,v)}{\|M(u,v)\|} + d^2 \cdot \frac{q(u,v)}{\|M(u,v)\|^2}, \tag{5.10}$$

where

$$\begin{aligned} M(u,v) &= S_u \times S_v, \\ p(u,v) &= \langle S_u \times M_v + M_u \times S_v, M \rangle, \\ q(u,v) &= \langle M_u \times M_v, M \rangle. \end{aligned}$$

Using an auxiliary variable: $\sigma = \|M(u,v)\| = \|S_u \times S_v\| > 0$, the inequality constraint $f(u,v) > 0$ can be converted to

$$\sigma^4 + d \cdot \sigma \cdot p(u,v) + d^2 \cdot q(u,v) > 0, \tag{5.11}$$

$$\sigma^2 = \|S_u \times S_v\|^2, \tag{5.12}$$

$$\sigma > 0. \tag{5.13}$$

These trivariate constraints are easier to test than other fourvariate conditions. By checking these constraints first, we have greatly accelerated the whole solution procedure.

5.4 Multivariate Solver for Constraints

We can solve a system of Equations (5.2)–(5.7) under the additional constraints of Equations (5.8)–(5.13) using the IRIT Library [41]. In Equations (5.6)–(5.7), the offset distance d appears in a squared form of d^2; thus, $-d$ will also be a solution whenever d is. Nevertheless, the negative offset $S(u,v) - d \cdot N(u,v)$ is invalid, and the redundant solutions from $-d$ should be filtered out.

The invalid negative offset solutions can be removed systematically by adding two auxiliary variables: $\sigma = \|S_u \times S_v\| > 0$ and $\tau = \|S_s \times S_t\| > 0$. Equations (5.6)–(5.7) are then replaced by the following four equations in lower degree:

$$2d \ \langle S(u,v) - S(s,t), S_u \times S_v \rangle = \sigma \|S(u,v) - S(s,t)\|^2, \tag{5.14}$$

$$2d \ \langle S(u,v) - S(s,t), S_s \times S_t \rangle = \tau \|S(u,v) - S(s,t)\|^2, \tag{5.15}$$

$$\sigma^2 = \|S_u \times S_v\|^2, \tag{5.16}$$

$$\tau^2 = \|S_s \times S_t\|^2. \tag{5.17}$$

Because of the solution procedure in a higher dimensional space, we need to spend more computing time in this approach (often three more times). Nevertheless, there is an improvement in numerical stability due to the reduction of degrees in Equations (5.14)–(5.15). One can decide which approach to take depending on the relative importance of efficiency and stability for specific applications.

Finally, note that even though we start the derivations of Equations (5.2)–(5.17) with the offset surface as well as the regular input surface, the final forms of these constraints are given only in terms of $S(u,v)$ and its first and second partial derivatives. This is a unique feature of our approach, which

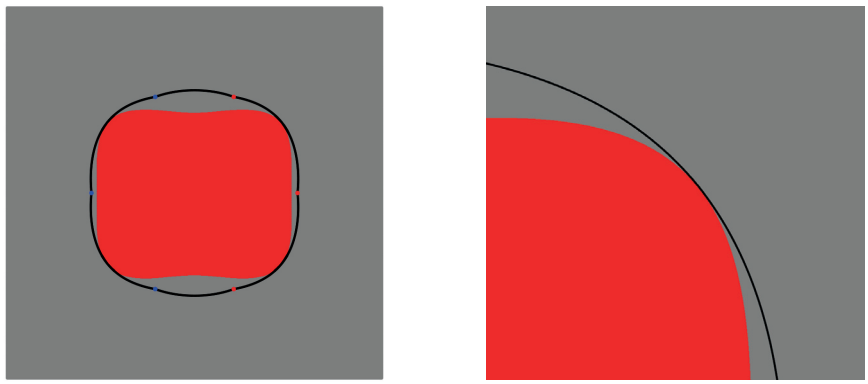


Figure 5.2: Curvature analysis on the surface $S(u, v)$: (a) the outer loop (in black) is the boundary of a redundant trimming region, whereas the red region is the set of (u, v)-parameters where $S(u, v)$ has one of the principal curvatures to the concave side larger than $1/d$; (b) a zoom-in view on the upper-right corner of the region.

may be very useful in the development of other algorithms for offset-related geometric problems.

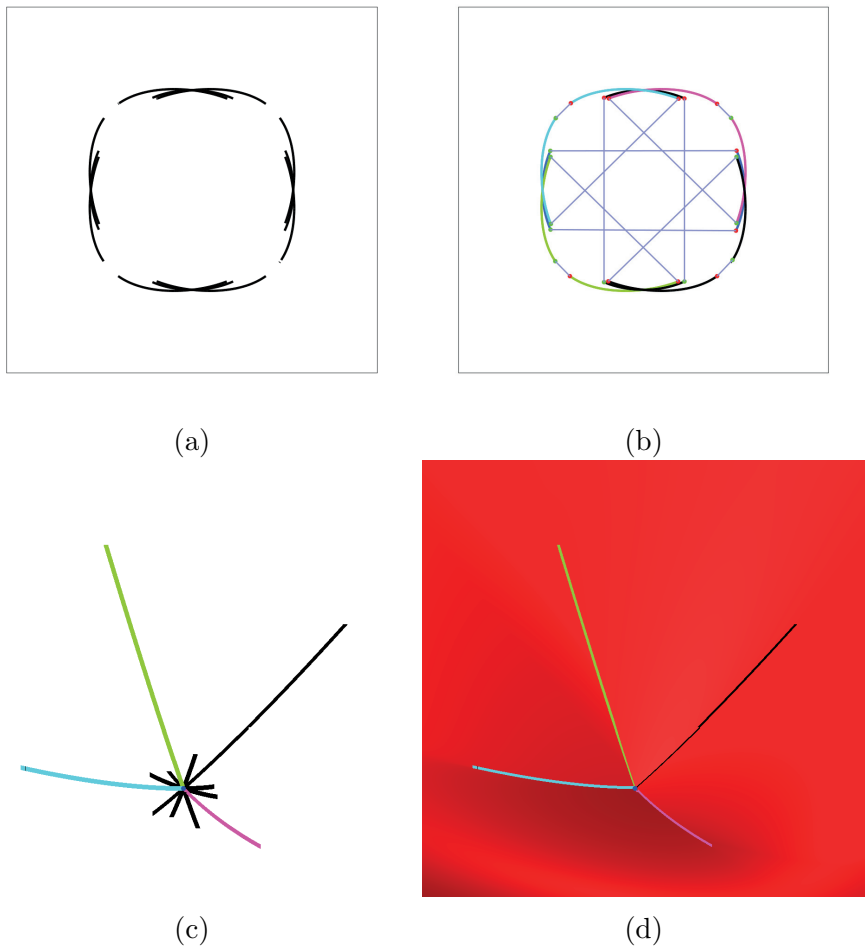


Figure 5.3: (a) An arrangement of solution curve segments from the constraint solver in the uv-domain; (b) the same curve arrangement, where matching segments are in the same color and matching endpoints are shown in blue line connections; (c) the arrangement of self-intersection curve segments in the xyz-space, where the color shows the correspondence with the matching uv-curve segments; (d) an X-junction with four branches of trimmed self-intersection curve segments on the offset surface (in the same color coding with (b) and (c)).

5.A Derivation of $f(u, v)$

We derive the formula of $f(u, v)$ as a function that depends on the first and second partial derivatices of $S(u, v)$:

$$\begin{aligned}
f(u,v) &= \langle O_u \times O_v, S_u \times S_v \rangle \\
&= \langle (S_u + d \cdot N_u) \times (S_v + d \cdot N_v), S_u \times S_v \rangle \\
&= \langle S_u \times S_v, S_u \times S_v \rangle + d \cdot \langle S_u \times N_v + N_u \times S_v, S_u \times S_v \rangle \\
&\quad + d^2 \cdot \langle N_u \times N_v, S_u \times S_v \rangle \\
&= \|S_u \times S_v\|^2 + d \cdot \langle S_u \times N_v, S_u \times S_v \rangle \\
&\quad - d \cdot \langle S_v \times N_u, S_u \times S_v \rangle + d^2 \cdot \langle N_u \times N_v, S_u \times S_v \rangle \\
&= \|S_u \times S_v\|^2 + d \cdot \langle S_u, N_v \times (S_u \times S_v) \rangle \\
&\quad - d \cdot \langle S_v, N_u \times (S_u \times S_v) \rangle + d^2 \cdot \langle N_u, N_v \times (S_u \times S_v) \rangle . \qquad (5.18)
\end{aligned}$$

For the sake of simplicity, we denote

$$M(u, v) = S_u(u, v) \times S_v(u, v), \qquad (5.19)$$

which is the (non-unit) normal of $S(u, v)$. Differentiating the unit-normal

$$N(u, v) = \frac{M(u, v)}{\|M(u, v)\|}, \qquad (5.20)$$

we get

$$N_u = \frac{M_u \, \|M\|^2 - M \, \langle M, M_u \rangle}{\|M\|^3}, \qquad (5.21)$$

$$N_v = \frac{M_v \, \|M\|^2 - M \, \langle M, M_v \rangle}{\|M\|^3}. \qquad (5.22)$$

Taking the cross product with M, they produce

$$N_u \times M = (M_u \times M)/\|M\|, \qquad (5.23)$$

$$N_v \times M = (M_v \times M)/\|M\|. \qquad (5.24)$$

By substituting Equations (5.23))–(5.24) to (5.18), we obtain

$$\begin{aligned}
f(u,v) &= \|M\|^2 + \frac{d}{\|M\|}\langle S_u, M_v \times M\rangle - \frac{d}{\|M\|}\langle S_v, M_u \times M\rangle \\
&\quad + \frac{d^2}{\|M\|}\langle N_u, M_v \times M\rangle \\
&= \|M\|^2 + \frac{d}{\|M\|}\langle S_u \times M_v - S_v \times M_u, M\rangle \\
&\quad - \frac{d^2}{\|M\|}\langle N_u \times M, M_v\rangle \\
&= \|M\|^2 + \frac{d}{\|M\|}\langle S_u \times M_v + M_u \times S_v, M\rangle \\
&\quad + \frac{d^2}{\|M\|^2}\langle M_u \times M_v, M\rangle \\
&= \|M\|^2 + d\cdot(p(u,v)/\|M\|) + d^2\cdot(q(u,v)/\|M\|^2),
\end{aligned}$$

where $p = \langle S_u \times M_v + M_u \times S_v, M\rangle$, and $q = \langle M_u \times M_v, M\rangle$.

5.B Relationship between $f(u,v)$ and Curvatures

$f(u,v)$ again can be written as follows.

$$\begin{aligned}
f(u,v) &= \|M\|^2 + \frac{d}{\|M\|}\langle S_u, M_v \times M\rangle - \frac{d}{\|M\|}\langle S_v, M_u \times M\rangle + \frac{d^2}{\|M\|}\langle N_u, M_v \times M\rangle \\
&= \|M\|^2 + \frac{d}{\|M\|}\langle S_u \times M_v - S_v \times M_u, M\rangle - \frac{d^2}{\|M\|}\langle N_u \times M, M_v\rangle \\
&= \|M\|^2 + \frac{d}{\|M\|}\langle S_u \times M_v + M_u \times S_v, M\rangle + \frac{d^2}{\|M\|^2}\langle M_u \times M_v, M\rangle \\
&= \|M\|^2 + d\cdot(p(u,v)/\|M\|) + d^2\cdot(q(u,v)/\|M\|^2),
\end{aligned}$$

where $p = \langle S_u \times M_v + M_u \times S_v, M\rangle$, and $q = \langle M_u \times M_v, M\rangle$.

First,

$$
\begin{aligned}
p(u,v) &= \langle S_u \times M_v + M_u \times S_v, M\rangle \\
&= \langle S_u \times M_v, M\rangle - \langle S_v \times M_u, M\rangle \\
&= \langle S_u \times (S_{uv} \times S_v + S_u \times S_{vv}), M\rangle - \langle S_v \times (S_{uu} \times S_v + S_u \times S_{uv}), M\rangle \\
&= \langle S_u \times (S_{uv} \times S_v), M\rangle + \langle S_u \times (S_u \times S_{vv}), M\rangle \\
&\quad - \langle S_v \times (S_{uu} \times S_v), M\rangle - \langle S_v \times (S_u \times S_{uv}), M\rangle \\
&= \langle S_{uv}(S_u \cdot S_v) - S_v(S_u \cdot S_{uv}), M\rangle + \langle S_u(S_u \cdot S_{vv}) - S_{vv}(S_u \cdot S_u), M\rangle \\
&\quad - \langle S_{uu}(S_v \cdot S_v) - S_v(S_v \cdot S_{uu}), M\rangle - \langle S_u(S_v \cdot S_{uv}) - S_{uv}(S_v \cdot S_u), M\rangle \\
&= \langle S_{uv}(S_u \cdot S_v), M\rangle - \langle S_{vv}(S_u \cdot S_u), M\rangle - \langle S_{uu}(S_v \cdot S_v), M\rangle + \langle S_{uv}(S_v \cdot S_u), M\rangle \\
&= fF\|M\| - Eg\|M\| - eG\|M\| + fF\|M\| = -\|M\|(Eg - 2fF + Ge).
\end{aligned}
$$

Second,

$$
\begin{aligned}
q(u,v) &= \langle M_u \times M_v, M \rangle \\
&= \langle M_v, M \times M_u \rangle \\
&= \langle M_v, M \times (S_{uu} \times S_v + S_u \times S_{uv}) \rangle \\
&= \langle M_v, M \times (S_{uu} \times S_v) \rangle + \langle M_v, M \times (S_u \times S_{uv}) \rangle \\
&= \langle M_v, S_{uu}(M \cdot S_v) - S_v(M \cdot S_{uu}) \rangle + \langle M_v, S_u(M \cdot S_{uv}) - S_{uv}(M \cdot S_u) \rangle \\
&= \langle M_v, -S_v(M \cdot S_{uu}) \rangle + \langle M_v, S_u(M \cdot S_{uv}) \rangle \\
&= -e\|M\| \langle M_v, S_v \rangle + f\|M\| \langle M_v, S_u \rangle \\
&= -e\|M\| \langle (S_{uv} \times S_v + S_u \times S_{vv}), S_v \rangle + f\|M\| \langle (S_{uv} \times S_v + S_u \times S_{vv}), S_u \rangle \\
&= -e\|M\| \langle (S_u \times S_{vv}), S_v \rangle + f\|M\| \langle (S_{uv} \times S_v), S_u \rangle \\
&= -e\|M\| \langle S_{vv}, S_v \times S_u \rangle + f\|M\| \langle S_{uv}, S_v \times S_u \rangle \\
&= -e\|M\| \langle S_{vv}, -M \rangle + f\|M\| \langle S_{uv}, -M \rangle \\
&= eg\|M\|^2 - f^2\|M\|^2 = \|M\|^2(eg - f^2).
\end{aligned}
$$

Finally,

$$
\begin{aligned}
f(u,v) &= \|M\|^2 + d \cdot \frac{p(u,v)}{\|M\|} + d^2 \cdot \frac{q(u,v)}{\|M\|^2} \\
&= \|M\|^2 - d(Eg - 2fF + Ge) + d^2(eg - f^2) \\
&= \|M\|^2(1 - 2d\frac{Eg - 2fF + Ge}{2\|M\|^2} + d^2\frac{eg - f^2}{\|M\|^2}) \\
&= \|M\|^2(1 - 2d\frac{Eg - 2fF + Ge}{2(EG - F^2)} + d^2\frac{eg - f^2}{EG - F^2}) \quad (where\ \|M\|^2 = EG - F^2) \\
&= \|M\|^2(1 - 2dH + d^2K).
\end{aligned}
$$

where $H = \frac{1}{2}\frac{eG - 2fF + gE}{EG - F^2}$ (mean curvature) and $K = \frac{eg - f^2}{EG - F^2}$ (gaussian curvature). Reminding that $H = (\kappa_1 + \kappa_2)/2$ and $K = \kappa_1 \cdot \kappa_2$ where κ_1 and κ_2 are principal curvatures of $S(u,v)$,

$$f(u,v) = \frac{\|M\|^2}{d^2}(\kappa_1 - \frac{1}{d})(\kappa_2 - \frac{1}{d}) \tag{5.25}$$

Hence, the inequality constraint (5.10) is equivalent to filtering out the region whether one of the principal curvatures is larger than the inverse of the offset radius, but not both.

5.3 Trimming Offset Surfaces

The solution curve segments (constructed by the constraint solver) in the $uvst$-space are projected to the uv-domain and generate an arrangement of planar curve segments. Because of the symmetry in the relations: $O(u,v) = O(s,t)$ and $O(s,t) = O(u,v)$, the projection to the st-domain should be the same as the one to the uv-domain. In practice, the two projections are slightly different in their computing results due to numerical error. When the difference is relatively large in some areas, we cut off these segments from the solution curve. The large difference hints on the unreliability of the result. The constraint solver often produces no solutions around the regions of numerical instability. Thus we start with an incomplete arrangement of solution curve segments.

We can explain the numerical instability geometrically using a curvature analysis as shown in Figure 5.2, where the closed loop (in black) is the boundary of a redundant trimming region for the test example of Figure 5.3. In other words, Figure 5.2 is the final result of the loop construction of Figure 5.3, starting from an incomplete curve arrangement of Figure 5.3 (a). The red region of Figure 5.2 is the set of (u,v)-parameters where $S(u,v)$ has one of its principal curvatures to the concave side larger than $1/d$. The boundary of this region corresponds to the singular curves of the offset surface $O(u,v)$,

where the offset self-intersection computation becomes highly unstable. One can notice that the loop (in black) has an almost tangential contact with the boundary of the red region, for which our constraint solver produces unreliable solutions which are discarded.

The missing parts usually correspond to the terminal points at the tips of some branches of the offset trimming curve in the xyz-space. We try to bridge the missing gap using an intersection curve tracing based on a sequence of pairs of osculating torii to the offset surface. At the limiting point very close to the terminal location, we are essentially approximating the offset surface using an almost identical tori, but with slightly different parameterizations. This approach closes the missing gap using the geometry of a limiting torus that approximates the offset surface within a small error bound.

The osculating tori are first constructed for the regular input surface, one for $S(u, v)$ and the other for $S(s, t)$, using their principal curvatures at the locations of (u, v) and (s, t), where (u, v, s, t) is the solution from the constraint solver for the endpoint of the solution curve that we are trying to extend. We use a construction scheme similar to Liu et al. [60], where the osculating torus is used for the acceleration of point projection. For intersection curve tracing, we use osculating tori of relatively small sizes and recompute them after advancing a short distance to reduce the error accumulation. The osculating tori for $O(u, v)$ and $O(s, t)$ are simply computed by offsetting the osculating tori for $S(u, v)$ and $S(s, t)$ by the offset distance d.

The hard part of the surface intersection at a singular intersection point is how to decide where to stop the intersection curve tracing. The terminal point is a singular intersection point, where the traced curve changes its direction abruptly to the opposite. In our case, the missing segment is incrementally

constructed from both sides of (u, v) and (s, t), and finally closed at the meeting point $(u, v) = (s, t)$, where the corresponding point (x, y, z) will be the terminal point of the self-intersection curve segment. The tori-intersection becomes unstable as the curve tracing approaches the terminal point, where the two osculating tori intersect almost tangentially. Thus instead of intersecting the two almost identical tori, we compute the intersection curve using their normal sections (by intersecting the tori with a plane determined by the curve tracing direction and an almost the identical normal vector of the two tori. We stop the curve tracing when the simultaneous tracings in (u, v) and (s, t) meet at a common location.

Once all the terminal points are computed and their (u, v) locations are detected, the gluing operation proceeds by matching two curve segments in the uv-domain to a connected branch in the xyz-space until a Y-junction, where the branch meets two other branches of the Y-junction. In the uv-domain, the tracing proceeds along two matching curve segments, each of which will meet a different uv-curve segment at a crossing location (u, v), where the corresponding offset point $O(u, v)$ is the exact location of the Y-junction. At the crossing location (u, v), the two uv-curve segments are cut, each into two pieces, and two redundant pieces are purged away. The tracing further proceeds along the remaining curve segment at the (u, v) crossing. Repeating the same step for the other matching curve (now interpreted as an st-curve), we can trace the other branch of the Y-junction.

Two closely located Y-junctions form an X-junction by shrinking the main branch to length zero and thus merging the two Y-junctions to a joint junction finally with four remaining branches (see Figure 5.3). In the tracing of curve segments in the uv-domain, an X-junction can be detected by the existence of

three uv-curve segments in the neighborhood of a (u, v)-crossing that the tracing encounters (see Figure 5.3(a)). It is sometimes difficult to decide whether it is just one X-junction or a pair of two Y-junctions, which is often a good indication to the chance of forming an X-junction.

5.4 Experimental Results

We have implemented our offset surface trimming algorithm using the proposed loop construction algorithm in C++ (Section 5.3 as well as the IRIT solid modeling library [41] for solving the system of equality and inequality constraints in Section 5.1), on an Intel Core i7-6700K 4.0GHz PC with a 32GB main memory. To demonstrate the effectiveness of our approach, we have tested our algorithm against several test examples of regular freeform surfaces.

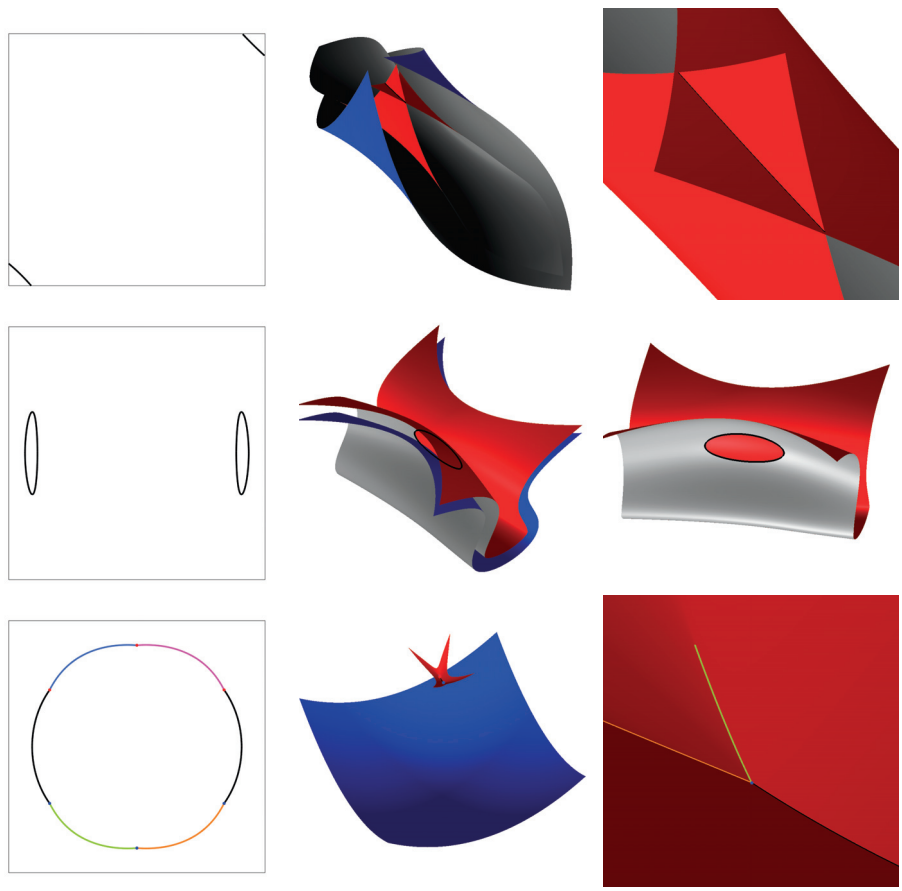
Figure 5.4 shows the results from testing our offset trimming algorithm on three examples of Maekawa et al. [62]. The first two test results are on easy cases, which shows that our algorithm works for general types of offset surfaces, not only specialized for handling near-singular offset self-intersections. In these two examples, the solution in the surface parameter domain has two separate components; clearly, one is the uv-component of the solution curve and the other is the st-counterpart of the solution. One can check the correctness of these solutions from the zoom-in views in the right column of Figure 5.4(c). The third example of Figure 5.4 is a typical case of offset surface trimming in the vicinity of near-singular offset self-intersections.

The color-coding in Figure 5.4(a) provides the matching information on the solution curve segments, where the pair of black segments (or loops in the

second example) corresponds to the main branch of the offset trimming curve (or the main loop in the second example). In the third example of Figure 5.4 (a), there are four other curve segments (shown in different non-black colors) on the trimming loop, each of which is a pair of matching segments with a common endpoint that corresponds to a terminal point at the tip of one branch on the offset trimming curve. In the zoom-in view of Figure 5.4(c), one can realize that it is hard to predict the exact location of the terminal point visually. Without having the correspondence map of the offset trimming loop in the uv-domain, it is extremely difficult to decide whether the curve tracing along the offset self-intersection in the xyz-space has under- or over-shooted the target location at the terminal point. Moreover, it is important to have a stable tracer along the offset self-intersection curve – the curve tracing will eventually reach a singular intersection point.

The constraint solver often produces an incomplete map for the offset trimming loop in the uv-domain. The first two examples of Figure 5.4 are the direct results from the constraint solver (computed with the formulae derived in Section 5.1). On the other hand, the third example is the final result of the offset trimming algorithm based on both Section 5.1 and 5.1, which is completed by removing all redundant segments and adding all missing segments to form a closed loop in the self-intersection curve. The correct detection of all terminal points is the main challenge of this work. Depending on the different levels of singularity at the terminal points of the offset self-intersection curve, there will always be some non-trivial cases where the offset trimming curve has very complex shapes.

To further test the performance of our method on more general types of offset surface trimming examples, we have generated two different sets of offset



(a) (b) (c)

Figure 5.4: Examples from Maekawa et al. [62]: (a) the arrangement of solution curves in the uv-domain; (b) the input surface $S(u, v)$ (in blue) and the offset surface $O(u, v)$ (in red), and the offset trimming curve (in black) in the xyz-space; (c) zoom-in view of (b).

surfaces: (i) scaling the input surface along the x-direction, and (ii) changing the offset distance. The progenitor surface $S(u, v)$ is a bicubic Bézier surface with 16 control points P_{ij}:

$$[P_{ij}] = \begin{bmatrix} (0, 0.4575, 0) & (0.1525, 0.4575, 0.305) & (0.305, 0.4575, 0.305) & (0.4575, 0.4575, 0) \\ (0, 0.305, 0.305) & (0.1525, 0.305, 0.7625) & (0.305, 0.305, 0.7625) & (0.4575, 0.305, 0.305) \\ (0, 0.1525, 0.305) & (0.1525, 0.1525, 0.7625) & (0.305, 0.1525, 0.7625) & (0.4575, 0.1525, 0.305) \\ (0, 0, 0) & (0.1525, 0, 0.305) & (0.305, 0, 0.305) & (0.4575, 0, 0) \end{bmatrix}$$

This surface is symmetric with respect to $u = 0.5$, $v = 0.5$, and $u = \pm v$, which is intended to produce an X-junction in the middle of the offset surface. We compute the offset trimming curve for the surface, and repeat the same computation for scaled versions of the surface along the x-direction, each with the offset distance fixed to $d = 0.15$. The offset trimming results are shown in Figure 5.5, where the leftmost two columns show the intermediate results from the constraint solver, and the rightmost two columns are the final results of the offset trimming construction. To show the correctness of the trimming curve construction on the offset surface, the zoom-in views on the most critical parts are also reported in Figure 5.6. As expected from the symmetry of the input surface, Figure 5.6 (a) demonstrates the construction of an X-junction with four branches, at the center of the offset surface. The last two examples belong to the case of having only one main branch with two singular endpoints, which can be detected by the missing segments in the incomplete loops shown in Figures 5.5(a)–(b).

Finally, we consider a variable offset distance d, and consider the structural evolution of the offset trimming curve as the distance d changes. As the input for this test, we take the fourth surface of Figure 5.5, the offset of which with the distance $d = 0.15$ had two Y-junctions, one long main branch and four short branches on the offset trimming curve. As we decrease the offset distance, the branches get shorter and shorter, and at some distance, only the

main branch remains. Even in the zoom-in views of Figure 5.7(d)–(i), it is hard to decide the branching structures (in the xyz-space) visually. Nevertheless, it is easy to tell the structures in their offset trimming loops in the uv-domain, as shown in Figure 5.7 (a). Five trimming curves, each with two Y-junctions, are shown in Figure 5.7 (b), whereas one curve with only one main branch is shown separately in Figure 5.7 (c).

5.5 Summary

We have presented a new approach to the offset surface trimming problem, which can deal with the generic nature of near-singular offset self-intersections by computing the branching structure in a stable way. The precise locations of the terminal points of the branches are extremely difficult to detect, in particular, since the offset surface has a considerably more complicated shape than the input surface. Nevertheless, by developing offset-free computing tools, we have made the whole construction procedure as stable as possible, mainly based on the regularity of the input surface. Even in some non-trivial cases where the branching structure is inevident even in the zoom-in view of small features (often enlarged by 10^4 times), the correspondence map of an offset trimming loop in the uv-domain stably clarifies the branching structure. In some sense, we have converted the near-singular structure of the problem to a regular arrangement of solution curves in the uv-domain.

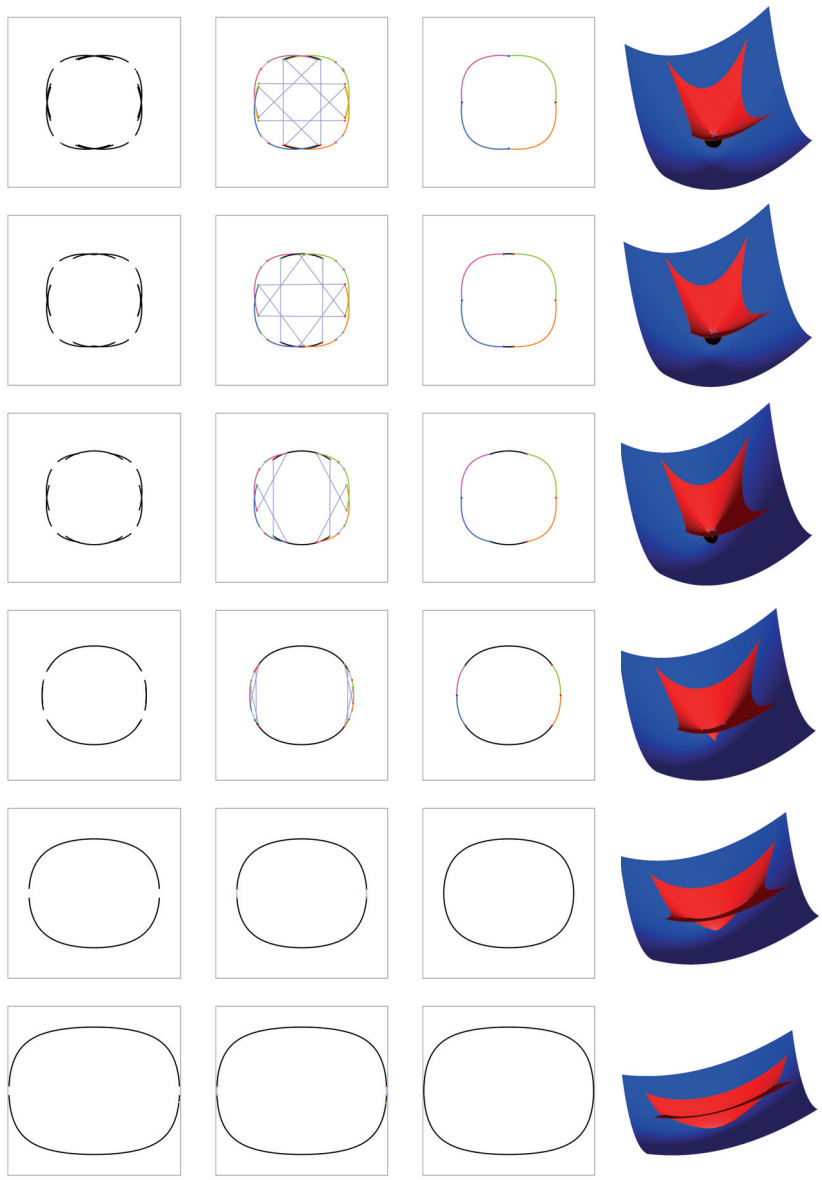


Figure 5.5: 1st row: x:y = 1:1, 2nd row: x:y = 1.01:1, 3nd row: x:y = 1.05:1, 4th row: x:y = 1.2:1, 5th row: x:y = 1.5:1, 6th row: x:y = 2:1. (a) Input surface $S(u, v)$; (b) offset surface $O(u, v)$; (c), (d) trimmed self-intersection curves on the offset surface (from (c): side-view and (d) bottom-view).

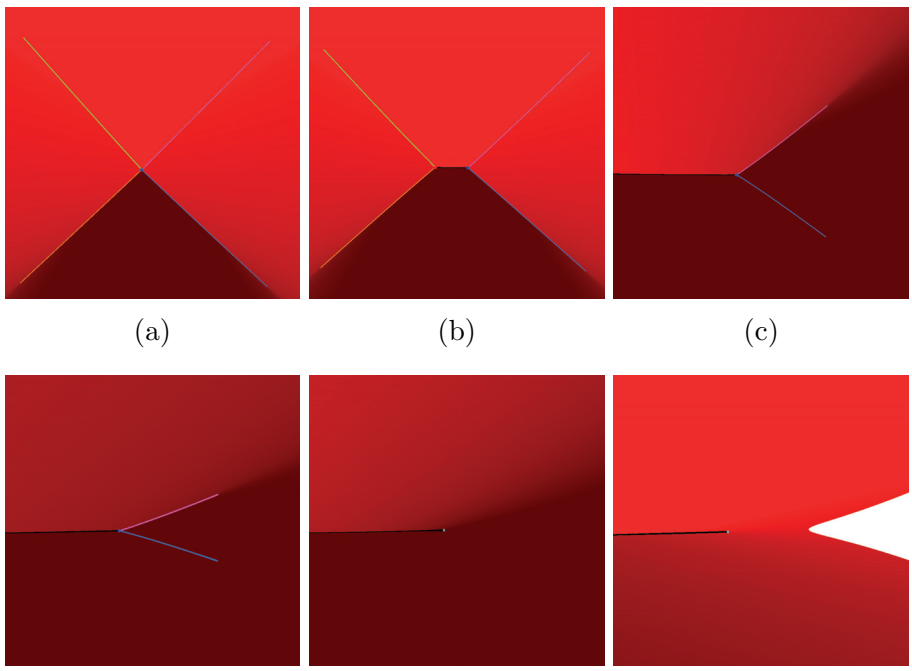


Figure 5.6: Zoom-in views of the offset trimming curves on the offset surface $O(u, v)$ in the xyz-space, where the control nets are scaled along the x-direction in the ratios of $x : y =$ (a) $1 : 1$, (b) $1.01 : 1$, (c) $1 : 05 : 1$, (d) $1.2 : 1$, (e) $1.5 : 1$, and (f) $2 : 1$, where the offset distance is fixed to $d = 0.15$. Cyan points in (e) and (f) represent detected singular points where $O(u, v) = O(s, t)$.

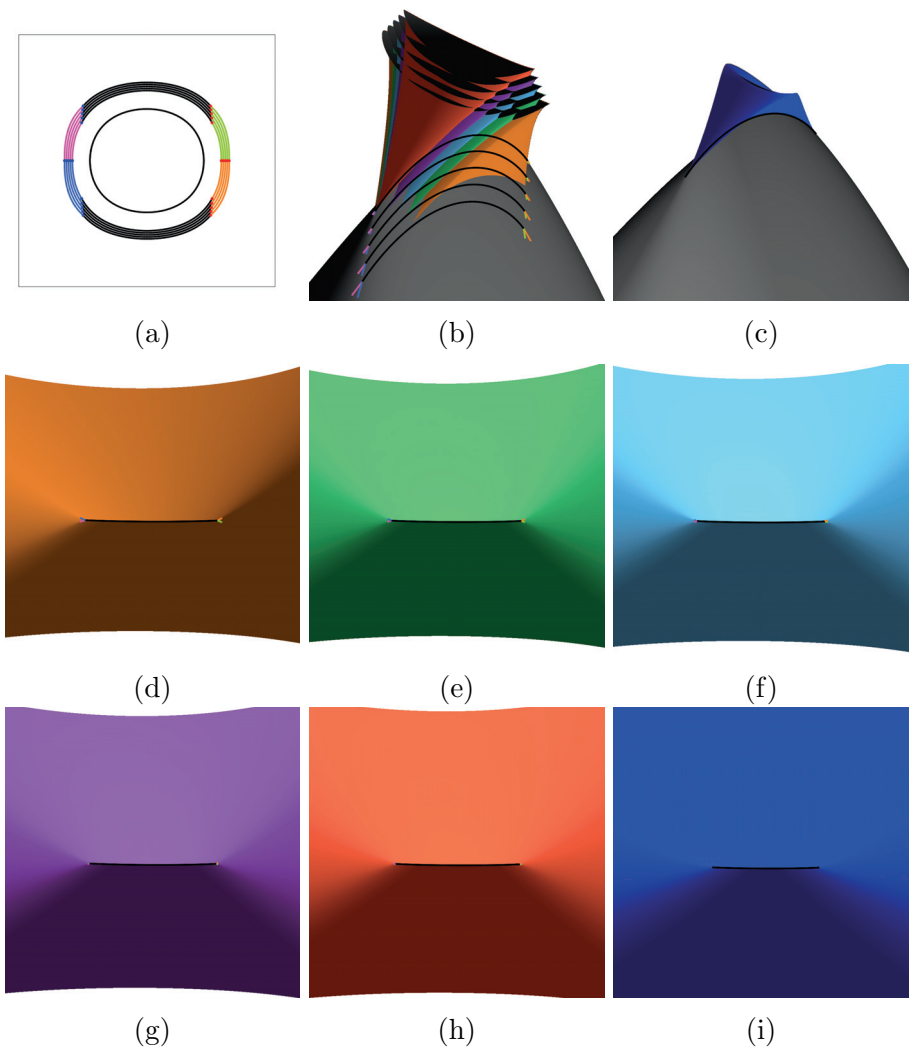
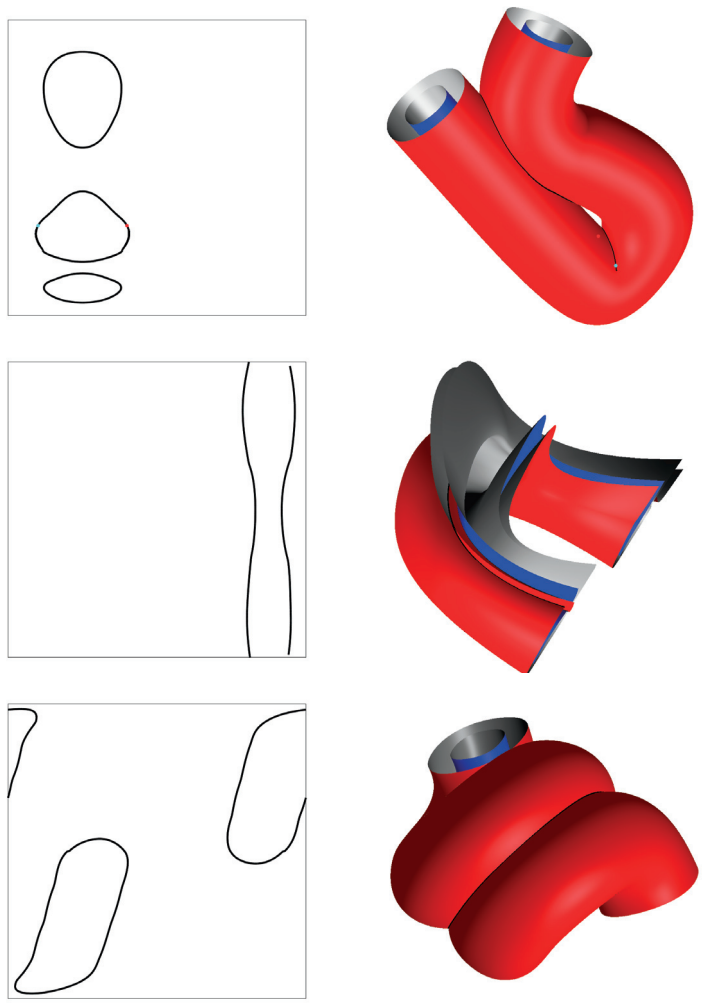


Figure 5.7: Offset trimming curves: (a) in the uv-domain and (b), (c) in the xyz-space; (d)–(i) zoom-in views of the offset trimming curves (when viewed from below toward the upward direction) on the concave side of the offset surfaces (with the offset radius d = (d) 0.1575, (e) 0.1545, (f) 0.1515, (g) 0.1485, (h) 0.1455, (i) 0.1200).



(a) (b)

Figure 5.8: Examples from Seong et al. [78]: (a): the arrangement of solution curves in the uv-domain; (b): the input surface $S(u, v)$ (in blue) and the offset surface $O(u, v)$ (in red), and the offset trimming curve (in black) in the xyz-space.

Chapter 6

Acceleration of trimming offset curves and surfaces

6.1 Motivation

Robustness is often one of the most important goals in designing a geometry processing tool. Here, "robust" can imply a variety of aspects: sometimes it means fast, numerically stable, computationally efficient, or resource-saving. In the typical CAD/CAM applications using offsets, attention is often on yielding precise shapes of the geometric objects. In NC machining, for instance, offsetting generates a path of a cutter tool that follows and cuts the hull of the object. Preciseness is more important than speed in this situation as designing the path is often a one-time process, and the precomputed path can be reused again and again to reproduce millions of products under the same milling machine. Speed and efficiency, however, become an essential factor in other types of offset applications. In particular, designing a fast and com-

putationally efficient offset trimming algorithm is required when handling a variety of offset radii with the same progenitor object or trimming offsets of continuously deforming objects.

In this section, we provide an alternative approach to identifying the self-intersections of offset curves and surfaces, which accelerate an offset trimming process. The new framework relies on a hierarchical spatial structure specialized in finding the self-collision and self-intersections of offset curves and surfaces. With the efficient data structure, we reduce the search space during self-intersection detection in offset curves and surfaces. We also reveal various geometric constraints to narrow down this search space further.

The main focus of Chapter 4 and 5 is on the identification of the correct branching structures and trimming regions in the self-intersections of offset curves and surfaces. There we skip discussions about the performance of the trimming algorithm intentionally. Before designing the acceleration algorithm, we first investigate the performance of the trimming algorithm proposed in the previous chapters.

The trimming algorithms in Chapter 4 and 5 first constructs implicit algebraic equations that constrain the self-intersections of offset curves or surfaces. The equations are formulated in the parametric space of curves or surfaces and solved with the subdivision-based multivariate equation solver. The solver subdivides NURBS curves or surfaces until the subdividends become flat enough to determine whether the constraints are satisfied, or the subdivision tolerance is reached.

Solving a set of constraint equations with the multivariate equation solver is a bottleneck of the offset trimming process, which makes the process slow due to several reasons. First, the subdivision used in solving equations is an ex-

pensive operation. When the progenitor curves and surfaces are Bézier curves or surfaces, for instance, the solver executes the *de Casteljau* subdivision algorithm. A single subdivision of a Bézier curve of degree n takes $O(n^2)$ interpolations, and that of a Bézier surface of degree $n \times n$ takes $O(n^3)$ interpolations. The computational cost increases exponentially in the dimension of parameters in the equation. These interpolations are even repeated during the recursive subdivision of the curves or surfaces.

The more critical problem that makes the trimming algorithm slow is that the trivial solutions such that $t = r$ or $(u, v) = (s, t)$ always exist in finding the self-intersections of the offset curves or surfaces. If the threshold of filtering the trivial solutions increases, the trimming algorithm will speed-up but may miss the subtle details in the self-intersections of the offsets. On the other hand, the small threshold value makes the equation solving problematically slow. The trivial solutions occur because of the independent parametrization of the duplicate progenitor surface used in the algebraic constraint equations for the self-intersections of the offset curves or surfaces. Every self-intersection point appears twice in the parametric domain as the solutions of the constraint equations are symmetric in the parametric space; if (t, r) is the solution of the self-intersection curve, so is (r, t). If (u, v, s, t) is the solution, so is (s, t, u, v).

Table 6.1 provides the execution time only spent in solving the constraint equations of the offset surface self-intersections using the IRIT system's subdivision-based multivariate geometric equation solving module. Most example surfaces we have tested are bicubic Bézier surfaces except **Maekawa_3** modeled with a bisextic Bézier surface and all of Seong's examples (**Pipe**, **SweepCurve** and **Helix**) modeled with B-spline surfaces. Experimental results show that the complexity of the self-intersection curves of the offset surfaces determines the

Example	Description	Time (hh:mm:ss)
Srf12_d80	$x : y = 12 : 10$, $d = 0.12$ (Fig. 5.7(i))	00:24:02
Srf12_d97	$x : y = 12 : 10$, $d = 0.1455$ (Fig. 5.7(h))	00:39:14
Srf12_d99	$x : y = 12 : 10$, $d = 0.1485$ (Fig. 5.7(g))	00:41:36
Srf12_d101	$x : y = 12 : 10$, $d = 0.1515$ (Fig. 5.7(f))	00:44:04
Srf12_d103	$x : y = 12 : 10$, $d = 0.1545$ (Fig. 5.7(e))	00:45:45
Srf12_d105	$x : y = 12 : 10$, $d = 0.1575$ (Fig. 5.7(d))	00:47:54
Srf10_d100	$x : y = 10 : 10$, $d = 0.15$ (Fig. 5.5(a))	01:10:31
Srf10.1_d100	$x : y = 10.1 : 10$, $d = 0.15$ (Fig. 5.5(b))	01:09:22
Srf10.5_d100	$x : y = 10.5 : 10$, $d = 0.15$ (Fig. 5.5(c))	00:59:09
Srf12_d100	$x : y = 12 : 10$, $d = 0.15$ (Fig. 5.5(d))	00:41:11
Srf15_d100	$x : y = 15 : 10$, $d = 0.15$ (Fig. 5.5(e))	00:30:50
Srf20_d100	$x : y = 20 : 10$, $d = 0.15$ (Fig. 5.5(f))	00:27:15
Maekawa_1	Maekawa's 1 (bell-shaped) (Fig. 5.4(a))	00:50:04
Maekawa_2	Maekawa's 2 (rolled sheet) (Fig. 5.4(b))	00:20:12
Maekawa_3	Maekawa's 3 (sextic bezier surface) (Fig. 5.4(c))	20:08:16
Pipe	Seong's (pipe) (Fig. 5.8(a))	01:07:36
SweepCurve	Seong's (curve swept surface) (Fig. 5.8(b))	00:42:34
Helix	Seong's (helix) (Fig. 5.8(c))	01:24:11

Table 6.1: Execution time of trimming offset surfaces based on the multivariate equation solving. d denotes the offset distance.

performance of the self-intersection detection algorithm when the degrees of the progenitor surfaces are the same. The fewer regions the constraint equa-

Description	Time (hh:mm:ss)
Srf10_d100, with the bicubic progenitor surface	01:10:31
Srf10_d100, with the bisextic progenitor surface	09:47:18
Srf20_d100, with the bicubic progenitor surface	00:27:15
Srf20_d100, with the bisextic progenitor surface	04:44:28

Table 6.2: The impact of the degree of the progenitor surfaces on the performace of offset trimming algorithm.

tions are tested and the more regions the auxiliary inequalities remove, the faster does the trimming algorithm yield the self-intersection curves. (See the simplicity of the self-intersection curves in Figure 5.7 (i), which only takes about 24 minutes to solve the constraint equations.)

Even though the offsets of bicubic surfaces demonstrate relatively short execution time, each example still takes from 20 minutes to about an hour in computing the solution of the constraint equations. Also, note that the bisextic Bézier surface in Figure 5.4 (c) takes almost 20 hours to compute the self-intersection curves of the offset surface, despite the simple structure of the self-intersections. To demonstrate the impact of the degree of the surface to the execution time of the equation solving, we compare the self-intersection computation time of bicubic Bézier surface offsets with bisextic Bézier surface offsets in Table 6.2. In Table 6.2, the degrees of example Bézier progenitor surfaces are raised from bicubic to bisextic without any modification in the shape of the surfaces. The degree elevation of the surfaces does not change the structure of the self-intersections of the offset surfaces, but make drastic increases in the execution time of the trimming algorithm. Because the degrees

of the polynomial terms in the constraint equations are squared when removing roots from the normal terms, the surfaces expressed in the higher degree terms worsen the performance of the trimming algorithm exponentially. Thus, it is crucial to reduce the number of subdivisions to make a faster trimming algorithm.

6.2 Basic Approach

The self-intersections of an offset curve (and an offset surface as well) are categorized into the local self-intersections and the global self-intersections, depending on the source of the intersections [46]. In the planar curve case, the local self-intersections of the offset curves occur when the curvature of the progenitor curves is larger than $1/d$, whereas the global self-intersections occur when two distant locations on the progenitor curves correspond to the same point on the offset curves. Kim et al. [46] first detect and eliminate the local self-intersections of the planar offset curve before handling the global self-intersections.

Similarly, the local and the global self-intersections also occur on offset surfaces; the only difference is that the local self-intersections of the offset surfaces are detected based on the principal curvatures of the progenitor surfaces. (See Section 5.3 for the details.) Nevertheless, the trimming algorithms proposed in Chapter 4 and Chapter 5 handle the local and the global self-intersections of offset curves and surfaces altogether through the unified algebraic equations. Subdivisions are always performed on the fly in tr-space (in case of trimming offset curves) or in $uvst$-space (in case of trimming offset surfaces), which require function evaluations in higher dimensions and higher degrees. This sit-

uation hinders the acceleration of trimming the self-intersections of the offset curves and surfaces. Therefore, the acceleration algorithm must distinguish the local self-intersections from the global self-intersections and minimize the number and the dimension of subdivisions during the computation.

In the geometry processing, a *hierarchical spatial structure* is a commonly used technique for the acceleration of the geometric queries and manipulations. A hierarchical spatial structure organizes a geometric object in 2D or 3D spatial bins of hierarchy where a bin in a higher level of the hierarchy contains all of its child bins below, each of which also contains its lower-level bins. Among the different types of hierarchical spatial structures, we employ the *bounding volume hierarchy*(BVH) to store curves and surfaces. The construction and the usage of the BVH are implemented in a divide-and-conquer manner: the complex geometric object is divided into smaller pieces, making the structure of the problem more straightforward and more comfortable to handle. The beauty of the BVH comes from the fact that it reduces the number of computations by bounding relatively complicated geometric objects with simpler objects and handling the queries on the simpler geometric objects instead. To find the self-intersections and trim the offset curves and surfaces, the BVH to enclose the target offset curves or surfaces are first constructed, and the self-intersections are detected using the BVH instead of the actual geometries.

In this chapter, we propose a special BVH and geometric queries operated on the BVH while trimming offset curves or surfaces. In the proposed BVH, the bounding volumes enclosing offset curves or surfaces are modeled from the composites of the bounding volumes of the progenitor curves or surfaces and the bounding volumes of their normals. Recall that an offset curve and surface

are defined as follows:

$$O(t) = C(t) + dN(t) \tag{6.1}$$

$$O(u, v) = S(u, v) + dN(u, v) \tag{6.2}$$

The normal term $N(t)$ (or $N(u, v)$) in the offset definition is also another curve or surface, so the normal term can be bounded by a spatial structure as well. Therefore, the maximum deviation between the actual offset geometry and its bounding volume, or a *bounding error* of the bounding volume, is the sum of the bounding error of the progenitor geometry and the bounding error of the normal geometry scaled by the offset distance d.

The challenges of the bounding volume construction arise from the fact that the offset bounding volumes easily become bulkier than those of the progenitor curves and surfaces. In particular, the bounding error of the normal terms becomes dominant when the offset distance is relatively large compared to the size of the progenitors. Fatter bounding volumes increase the amount of the local space search in the detection of the offset self-intersections and deteriorate the performance of the trimming algorithm as well. We also propose several trimming techniques to reduce the number of comparisons in the intersection tests. With these techniques, the local bounding volume pairs that do not contribute to the self-intersections of offset curves and surfaces are eliminated in the early stages of the trimming algorithm.

In the following sections, we first briefly demonstrate how to construct the BVH of the offset curves from the combination of the BVH of the progenitor curves and the BVH of the normals. We also show how to employ the BVH in trimming the offset curves. We then propose a method to construct the BVH of the offset surfaces and to trim the offset surfaces using the BVH. The

BVH of offset curves is mentioned for the completeness of the justification of the BVH-based trimming algorithm so that we will give more attention to the construction of the offset surface BVH. Furthermore, the performance gain of using the BVH will be much more significant in trimming the offset surfaces.

6.3 Trimming an Offset Curve using the BVH

Equation 4.4 reveals the isosceles relation between the self-intersection points on an offset curve and their corresponding progenitor points. Among the self-intersection points that we obtain from the solution of the constraint equations, the self-intersection points which belong to the local self-intersections of the offset curve are eliminated with the redundant curve segments in the post-process of the equation solving. We modify this algorithm by replacing the constraint equation solving with the geometric intersection tests on the bounding volumes enclosing the offset curve. Here the bounding volumes must be designed such that the intersection tests take less computational effort but still guarantee the accuracy of the results.

Before explaining the details of the BVH-based offset curve trimming algorithm, we suppose that the progenitor curve $C(t)$ be a planar curve $(x(t), y(t))$ where $x(t)$ and $y(t)$ are either cubic Bézier polynomials or cubic B-spline polynomials. When the latter is used, the curve is converted to a set of piecewise cubic Bézier forms through the knot insertion beforehand.

Each offset curve is enclosed by a set of bounding volumes, each of which encloses a monotone curve segment in the offset curve. We select Axis-Aligned Bounding Boxes (AABBs) as the bounding volumes of the offset curves because the coordinate-wise addition can be used in the construction of the

bounding volumes. Here the monotonicity of a curve segment in a leaf AABB is important because the AABB of a monotone curve is easily constructed by comparing only the endpoints of the curve. The monotonicity condition also assures that two curve segments intersect at most at a single point when their AABBs overlap. The intersection points are then easily computed by applying a point projection method of Hu and Wallner [39] or other Newton iteration methods on the curve segments enclosed in the colliding bounding volumes.

To divide an offset curve into a set of monotone curve segments, the extreme points of $O(t)$ must be identified in advance. Let $O(t) = (O_x(t), O_y(t))$ be an offset curve of $C(t) = (x(t), y(y))$ where $O(t) = C(t) + dN(t)$. Then the derivatives of $O(t)$ are formulated as follows.

$$\begin{aligned} O'(t) &= C'(t) + dN'(t) \\ &= (x'(t), y'(t)) + d\ \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}}(x'(t), y'(t)) \\ &= (x'(t), y'(t))(1 + d\ \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}}) \\ &= C'(t)(1 + d\kappa(t)) \qquad (6.3) \\ O''(t) &= C''(t)(1 + d\kappa(t)) + C'(t)d\kappa'(t) \\ & \qquad (6.4) \end{aligned}$$

Extreme points to separate the curve to the monotone curve segments include x-extreme points, y-extreme points and inflection points. Using Equation (6.3) and (6.4), the extreme points of the offset curve are identified as the solutions

of the following equations.

$$
\begin{aligned}
O'_x(t) &= x'(t)(1 + d\kappa(t)) = 0 \quad \text{(x-extreme point)} \\
O'_y(t) &= y'(t)(1 + d\kappa(t)) = 0 \quad \text{(y-extreme point)} \\
O'_x(t)O''_y(t) - O'_y(t)O''_x(t) &= (x'(t)y''(t) - y'(t)x''(t))(1 + d\kappa(t))^2 = 0 \quad \text{(inflection point)}
\end{aligned}
\tag{6.5}
$$

Equation (6.5) implies that the extreme points of the offset curve occur at the same t as those of the progenitor curve in addition to the singular points of the offset curve where the curvature becomes $-\frac{1}{d}$. Singular points of the offset curve are dependent on the offset distance d, whereas the extreme points of the progenitor curve can be shared between the offset curves with varying offset distance.

The computational cost of finding the exact singular points in an offset curve is high because one has to solve the following equation.

$$
\kappa(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}}(x'(t), y'(t)) = -\frac{1}{d} \tag{6.6}
$$

Even for a simle cubic Bézier curve, Equation (6.6) includes the polynomial terms of degree 12. Instead of directly solving Equation 6.6 for the entire domain of the curve, we do simple monotonicity tests to figure out whether the monotonicity of the curve segment is guaranteed or not. The segments not guaranteed to be monotone are subdivided into smaller curve segments and tested for the monotonicity again. The performance of the algorithm depends on the number of subdivisions: if we can find the monotone curve segment in the early stages during subdivision, it will reduce the number of intersection tests between the bounding volumes as well.

The monotonicity of the offset curve is tested in the following steps. We start with the offset curve separated by x-extreme and y-extreme and inflection

points of the progenitor curve. Extreme points of the progenitor curve can be easily computed. (For instance, we only need to solve at most quadratic equations for a cubic progenitor curve) For an x-monotone, y-monotone, and inflection-free progenitor curve segment, its corresponding offset curve segment is checked to have any singular points in it. To this intent, we first check the minimum value of $x'(t)y''(t) - x''(t)y'(t)$ in the curve segment: if this value and the offset distance d are positive, we know that $O'(t) \neq 0$ for this curve segment, which makes the segment monotone. If the minimum value is negative, we further investigate the range of $\kappa(t)$ by computing the nominator and the denominator of $\kappa(t)$. If we can guarantee the minimum value of $\kappa(t)$ is larger than $-\frac{1}{d}$, we conclude that the curve segment is monotone and stop the subdivision of the segment.

The monotonicity test yields a hierarchy of the bounding volumes, each of which encloses the monotone offset curve segment. We perform the local self-intersection tests and the global self-intersection tests on this BVH. The global self-intersection tests are performed by comparing whether two bounding volumes in BVH overlap or not. Sorting the bounding volumes along the x-axis (or the y-axis) in 2D further accelerates the speed of the tests as it reduces the number of comparisons between the bounding volumes.

Figure 6.1 shows the trimmed offset curves computed using the proposed BVH. Bounding volumes generated during offset curve trimming are also shown as red boxes in Figure 6.2. In Figure 6.2, the bounding volumes in the concave side of the offset curves and near the singularities are small compared to those in the convex side. This is because the curvature tests based on finding the range of $\kappa(t)$ can quickly prune the BVH in the convex side of the offset curves but must search the BVH further in the concave side where the

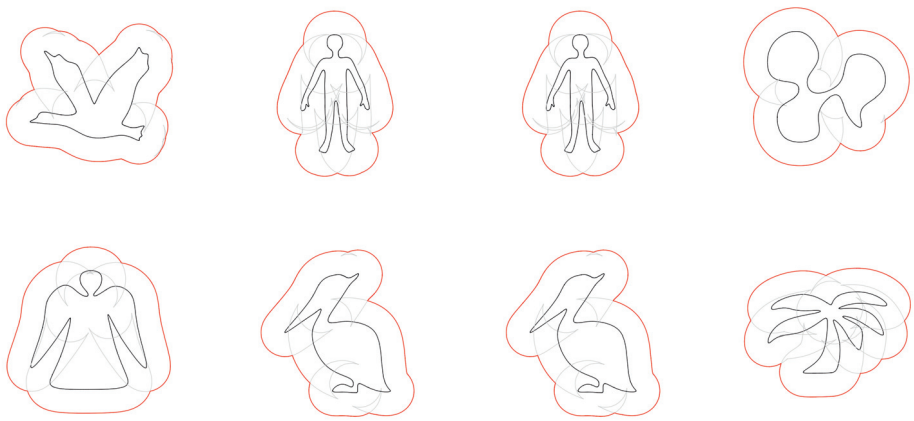


Figure 6.1: Examples of trimmed offset curves.

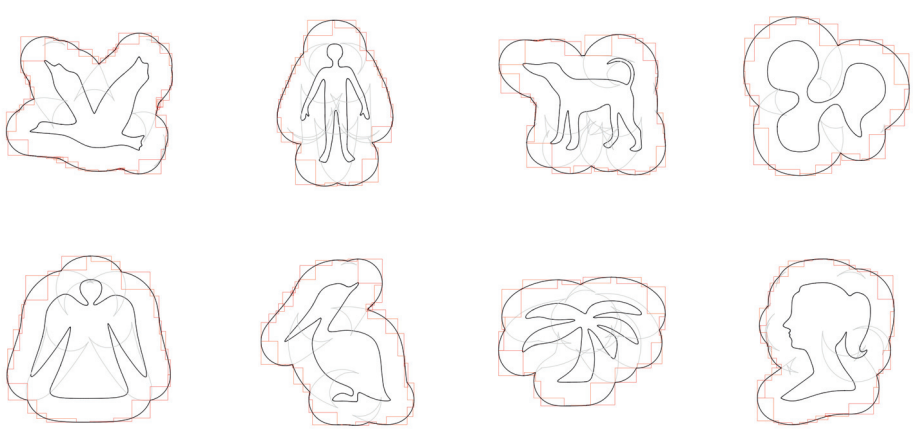


Figure 6.2: Bounding volumes enclosing offset curves. Axis-aligned bounding boxes(AABBs) are used to bound monotone offset curve segments.

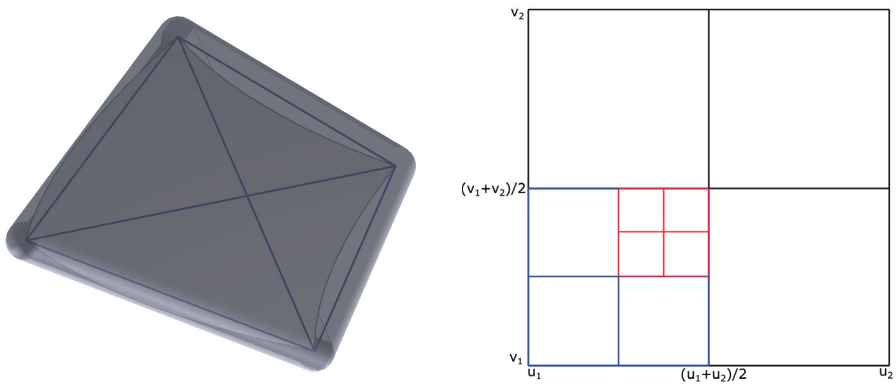
sign of $\kappa(t)$ approaches to $-\frac{1}{d}$. However, the overall number of subdivisions is still low compared to solving the constraint intersection equations for the entire domain of the curves, as proposed in Chapter 4.

6.4 Trimming an Offset Surface using the BVH

6.4.1 Offset Surface BVH

Similar to an offset curve, we enclose an offset surface with a hierarchical spatial structure of the bounding volumes. Each bounding volume must be a simple-shaped object that fits the acceleration of the decision tests and the geometric operations used in detecting the self-intersections and removing the redundant regions from the offset surface. Previous work on the BVH has proposed various types of bounding volumes enclosing the general surface. Among those bounding volumes, spheres, axis-aligned bounding boxes (AABBs), oriented bounding boxes (OBBs) are widely-used bounding volumes because they are easy to construct and maintain. However, those bounding volumes are often too loose relative to the actual geometry. Swept sphere volumes such as line swept spheres (LSS), and rectangle swept spheres (RSS), on the other hand, are more sophisticated bounding volumes that give a more tight fit to the actual geometry [54]. In using swept sphere volumes, we must design a simple-shaped geometry approximation (e.g., a line for LSS and a rectangle for RSS) and identify the bounding error, or the thickness of the bounding volume, which is the maximum difference between the geometry approximation and the actual geometry enclosed.

In this section, we design a new BVH enclosing the offset surface that is also specialized in the decision tests performed in offset surface trimming. The



(a) (b)

Figure 6.3: (a) An example offset tetrahedron enclosing the surface. (b) Recursive subdivion of the domain in building the BVH of the surface.

bounding volume in the new BVH is a kind of a tetrahedron swept sphere, or an *offset tetrahedron*. The hierarchy of bounding volumes is constructed recursively from a root node enclosing the offset surface of the entire domain to leaf nodes enclosing only the fraction of patches on the offset surface. Let $O(u, v)$ be the offset surface of the progenitor surface $S(u, v)$ defined on $[u_1, u_2] \times [v_1, v_2]$. To build the BVH, we split the offset surface along u and v-iso lines of the surfaces and yield the subpatches of the surface, the domains of which are $[u_1, \frac{u_1+u_2}{2}] \times [v_1, \frac{v_1+v_2}{2}]$, $[\frac{u_1+u_2}{2}, u_2] \times [v_1, \frac{v_1+v_2}{2}]$, $[u_1, \frac{u_1+u_2}{2}] \times [\frac{v_1+v_2}{2}, v_2]$, and $[\frac{u_1+u_2}{2}, u_2] \times [\frac{v_1+v_2}{2}, v_2]$ as shown in Figure 6.3 (b). The bounding volume of $O(u, v)$ contains a tetrahedron formed by $O(u_1, v_1)$, $O(u_2, v_1)$, $O(u_2, v_2)$ and $O(u_1, v_2)$ which are four corner points of the offset surface and spheres swept along the hull of tetrahedron. A radius of the sweeping sphere is the maximum difference between the tetrahedron and the actual offset surface. The

radius is determined so that the sphere-swept tetrahedron must contain all of the bounding volumes enclosing the subpatches of the surface. The subpathes recursively split, as shown in Figure 6.3 (b) until the length of both u and v domains of the patch becomes smaller than the predefined tolerance. When the subpath cannot split anymore, the bounding volume enclosing the patch becomes a leaf node of the offset surface BVH.

Whereas the structure of bounding volumes is constructed from the top-most root node enclosing the whole offset surface down to leaf nodes, the bounding error of each offset tetrahedron is computed in the opposite direction: the bounding error of any non-leaf offset tetrahedron is computed from the bounding errors of their four child offset tetrahedra. The question is then how to compute a bounding error of a leaf node that guarantees to enclose the actual offset surface patch. We will explain how to compute the bounding error of the leaf node and how to derive the bounding errors of the non-leaf nodes from their child nodes below.

Bounding Error of a Leaf Offset Tetrahedron

The bounding error of a leaf node must be large enough to guarantee that the offset tetrahedron encloses the actual offset surface patch, but the large bounding error also makes the bounding volume too bulky, which hinders the acceleration of the offset trimming algorithm. In the previous work on offset tetrahedron-based BVHs, Kim et al. [45] represent the NURBS surface patch in a leaf node with the simpler NURBS surface such as the Coons patch. To construct a BVH for a semi-regular quad mesh, Kang et al. [42] model a leaf patch as a bilinear surface. When the surface is parametrized in terms of the rational polynomial functions, the natural choice of the underlying

representation of a leaf bounding volume would be the rational polynomial surface of the lower degree because of the easiness in handling the upper bound of the bounding error. For those geometries, the bounding error between the actual geometry and the bounding volume can also be represented in terms of the rational polynomial functions.

If we model a leaf node of the offset surface BVH by approximating the offset surface with another NURBS representation $X(u, v)$ of lower degree, the bounding error between $O(u, v)$ and $X(u, v)$ becomes ϵ as follows:

$$|O(u, v) - X(u, v)| = |S(u, v) + dN(u, v) - X(u, v)| \leq \epsilon \tag{6.7}$$

We can also interpret ϵ as a Hausdorff distance between $O(u, v)$ and $X(u, v)$, which is the maximum value of the minimum distance values between $O(u, v)$ and $X(u, v)$ for all of the points on the surface. Unfortunately, it is difficult to compute ϵ because $N(u, v)$ is not rational in general and makes the degree of Inequality (6.7) too high. Instead of approximating the leaf node with NURBS representation directly, we construct the bounding volume of the offset surface with simpler geometry. By approximating the progenitor surface $S(u, v)$ with another type of geometry that is simpler to compute and bound the normal of the surface, we make the process of computing the bounding error of the offset surface approximation less complicated.

A torus patch is what we propose to approximate each progenitor surface patch $S(u, v)$ and use to construct the leaf bounding volume of the offset surface BVH. To construct a torus patch for the given surface patch, we adopt the method of Lie et al. [60], as mentioned in Section 5.3. This method finds the osculating torus at a point on the surface as follows: The major radius R and the minor radius r of the torus are computed from the principal curvatures

κ_1 and κ_2 at a point on the surface such that $R = \frac{1}{\kappa_1}$ and $r = \frac{1}{\kappa_1} - \frac{1}{\kappa_2}$. The torus is aligned so that the principal directions of the surface and the torus patch match.

The torus patch is advantageous in handling the offset operations of the surface because the torus is closed under offsetting. Let $T(s,t)$ be a torus where $s \in [-\pi, \pi]$ and $t \in [-\pi, \pi]$. Then $T(s,t)$ and the normal field $N_t(s,t)$ of $T(s,t)$ are given as follows.

$$T(s,t) = ((R + r\cos t)\cos s, (R + r\cos t)\sin s, r\sin t) \tag{6.8}$$

$$N_t(s,t) = \frac{T_s(s,t) \times T_t(s,t)}{|T_s(s,t) \times T_t(s,t)|} = (\cos t \cos s, \cos t \sin s, \sin t) \tag{6.9}$$

When $T(s,t)$ is offset by the distance d, the offset torus $T_o(s,t)$ becomes

$$\begin{aligned} T_o(s,t) &= T(s,t) + dN_t(s,t) \\ &= ((R + r\cos t)\cos s, (R + r\cos t)\sin s, r\sin t) + d(\cos t \cos s, \cos t \sin s, \sin t) \\ &= ((R + (r+d)\cos t)\cos s, (R + (r+d)\cos t)\sin s, (r+d)\sin t). \end{aligned} \tag{6.10}$$

Therefore, the offset of torus is another torus of which minor radius increases by the offset distance d while having the same major radius R.

Fitting the progenitor surface with torus patches makes the bounding error analysis of the offset surface easier. From Equation (6.10), we know that the osculating torus of a point on the offset surface is easily derived from the osculating torus of the corresponding point on the progenitor surface by merely adding the offset distance d to the minor radius of the torus. The maximum deviation between the offset surface and the osculating torus of the offset surface is then represented as a combination of the maximum deviation of the position fields and the maximum deviation of the normal fields between the progenitor surface and its osculating torus.

Suppose $S(u,v)$ be the progenitor surface and $T(s,t)$ be the osculating torus at some point (u_0, v_0) on $S(u,v)$. We denote ϵ_{torus_pos} be the maximum bounding error between $T(s,t)$ and $S(u,v)$ and ϵ_{torus_nor} the maximum bounding error between $N_t(s,t)$ and $N(u,v)$ as follows:

$$|T(s,t) - S(u,v)| \leq \epsilon_{torus\_pos} \tag{6.11}$$

$$|N_t(s,t) - N(u,v)| \leq \epsilon_{torus\_nor} \tag{6.12}$$

where $N_t(s,t)$ and $N(u,v)$ are the normal fields of $T(s,t)$ and $S(u,v)$, respectively. The maximum bounding error between the offset surface $O(u,v)$ and its osculating torus $T_o(s,t)$ satisfies the following inequality.

$$\begin{aligned} |O(u,v) - T_o(s,t)| &= |S(u,v) + dN(u,v) - T(s,t) - dN_t(s,t)| \\ &\leq |S(u,v) - T(s,t)| + d|N(u,v) - N_t(s,t)| \\ &\leq \epsilon_{torus\_pos} + d\epsilon_{torus\_nor} \end{aligned} \tag{6.13}$$

By approximating the progenitor surface patches with the osculating tori, the offset surface also gains the bounding error derived from the bounding errors of the progenitor surface. In the experiments, we fit each surface patch with the torus osculating at the center point $S(u_{mid}, v_{mid}) = S(\frac{u_1+u_2}{2}, \frac{v_1+v_2}{2})$ when the domain of the surface is $[u_1, u_2] \times [v_1, v_2]$.

Figure 6.4 shows a surface patch on the progenitor surface, the corresponding offset surface patch, and their osculating tori. In Figure 6.4 (a), the grey surface represents the progenitor surface $S(u,v)$, and the patch marked by a red quadrangle is the surface patch for which we want to find the leaf bounding volume. This patch is approximated by a torus patch osculating at $S(u_{mid}, v_{mid})$. As the osculating torus only gives a local fitting near the osculating point $S(u_{mid}, v_{mid})$, we cut out the torus by projecting the boundaries

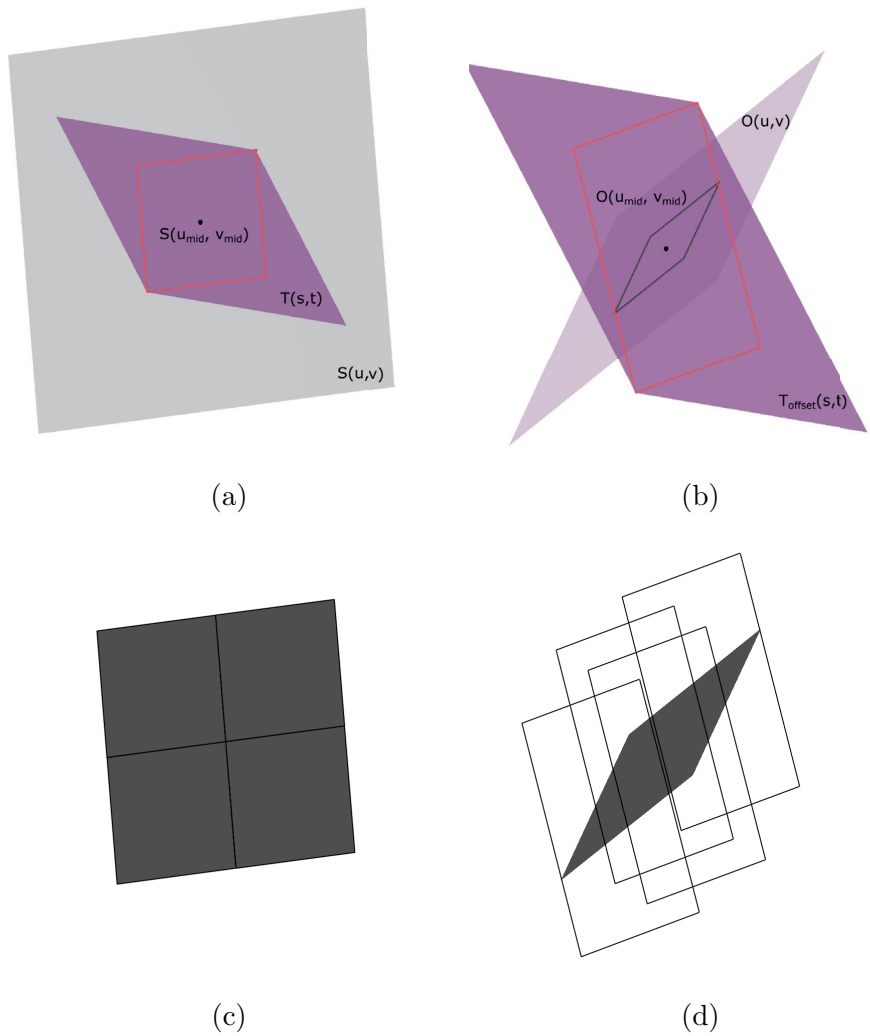


Figure 6.4: (a) Oscualting tori at the progenitor surface. (b) Offset of torus is also the osculating torus of the offset surface. 4 adjacent toroidal patches of progenitor surface (c) and their counterparts on offset surface (d).

of the surface patch onto the torus and generate a torus patch bounded by four corner points $T(s_1, t_1)$, $T(s_2, t_2)$, $T(s_3, t_3)$ and $T(s_4, t_4)$. The original surface patch and the bounded torus patch almost coincide in Figure 6.4 (a) This torus patch is bounded again by slightly bigger torus patch bounded by the s- and t-isolines of the osculating torus, $T(\min s_i, t)$, $T(\max s_i, t)$, $T(s, \min t_i)$ and $T(s, \max t_i)$ to simplify the computation (marked as the purple surface in Figure 6.4 (a)).

Figure 6.4 (b) shows the corresponding offset surface patch and its osculating torus. Here, the actual offset surface $O(u, v)$ is marked as light purple, whereas the leaf surface patch of interest is marked as a grey quad. Red rectangle represents the torus patch that has four corner points $T_o(s_1, t_1)$, $T_o(s_2, t_2)$, $T_o(s_3, t_3)$ and $T_o(s_4, t_4)$ and the dark purple-colored patch represents the torus patch bounded by the same s- and t-isolines as the torus patch in Figure 6.4 (a). In Figure 6.4 (c), we observe that the boundaries of the surface patches and their torus approximations coincide, and there are few overlaps between the neighboring torus patches. Those boundaries, however, no longer coincide well in the offset surfaces, as shown in the discrepancies of the boundaries in Figure 6.4 (b). Because of these discrepancies, the torus patches bounding the leaf nodes of the offset surface overlap severely with each other, as shown in Figure 6.4 (d).

The severe overlaps between nearby torus patches make it difficult to compute the bounding error of the parent torus patch that encloses their child torus patches. Each torus patch is constructed independently on the certain point on the offset surface, but an aggregate of child nodes can have many different geometric configurations depending on the local geometry, causing the computation of bounding errors complicated. Instead of directly constructing

a hierarchy from torus patches, we bound again a torus patch with an offset tetrahedron where the tetrahedron comes from four corner points of the offset surface, $O(u_1, v_1)$, $O(u_2, v_1)$, $O(u_1, v_2)$ and $O(u_2, v_2)$. Offset tetrahedra formed by the points on $O(u, v)$ are more convenient in building up the hierarchy of the bounding volumes because they can share the corner points with the adjacent tetrahedra. Briefly speaking, the offset surface patch in a leaf node is bounded by a sphere-swept tetrahedron, which is again bounded by the osculating torus of the offset surface. Figure 6.5 depicts this two staged bounding volume construction. The figures are drawn schematically such that the surfaces are simplified as the curves, the osculating tori as the arcs, and the tetrahedra as the lines. However, the relation shown in Figure 6.5 can also be applied to the case of bounding the offset surface, without loss of generality.

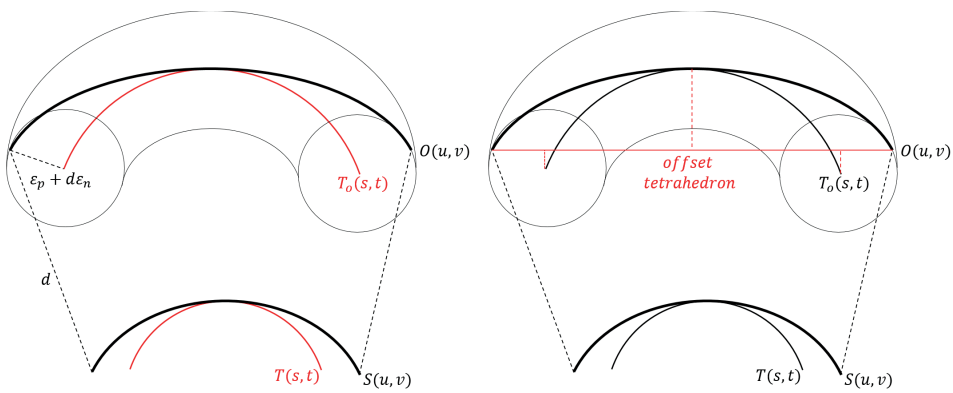


Figure 6.5: Schematic diagram of the bounding error computation using the offset osculating tori.

From the discussion above, we know that the maximum distance between the offset torus and the offset surface is $\epsilon_{torus\_pos} + d\epsilon_{torus\_nor}$. Then the max-

imum distance between the offset tori and the tetrahedron is the addition between the maximum distance between the offset tori and the offset surface and the maximum distance between the offset surface and the tetrahedron as follows.

$$\begin{aligned}
|T_o(s,t) - Tet(x,y,z)| &= |T(s,t) + dN_t(s,t) - Tet(x,y)| \\
&= |T(s,t) + dN_t(s,t) - O(x,y) + O(x,y) - Tet(x,y)| \\
&\leq |T(s,t) + dN_t(s,t) - O(x,y)| + |O(x,y) - Tet(x,y)| \\
&\leq (\epsilon_{torus_pos} + d\epsilon_{torus_nor}) + \epsilon_{offset_tetra} \qquad (6.14)
\end{aligned}$$

where $Tet(x,y,z)$ is a tetrahedron constructed from four points on $O(u,v)$. From the equation 6.13 and 6.14, we determine the bounding error of the offset tetrahedron as follows.

$$\epsilon \leq (\epsilon_{torus\_pos} + \epsilon_{torus\_nor}) \times 2 + \epsilon_{offset\_torus} \qquad (6.15)$$

We defer the explanation of the details of computing the actual error values, ϵ_{torus_pos}, ϵ_{torus_nor} and ϵ_{offset_torus} in the discussion above. To compute these values, we must find the maximum values of the lefthand side of the inequalities 6.11, 6.12 and 6.13. These computations can be done algebraically by differentiating the lefthand side equation with respect to u, v, s, and t. The computational cost, however, will be prohibited because the equations to solve consists of a combination of rational and non-rational terms from $S(u,v)$ and $N(u,v)$, as well as the transcendental terms from $T(s,t)$. Instead of finding the upper bound of the inequalities algebraically, we exploit the geometric properties of the torus to simplify the bounding errors ϵ_{torus_pos}, ϵ_{torus_nor} and ϵ_{offset_torus}.

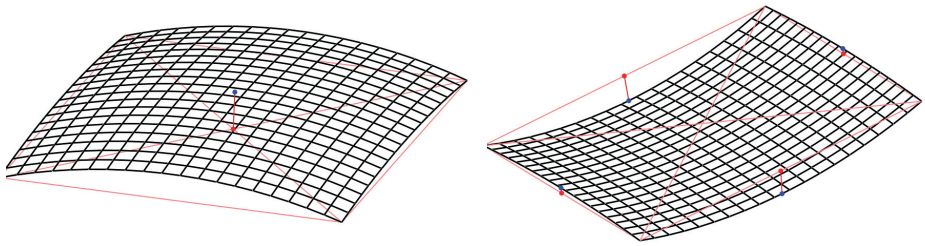
First, ϵ_{torus_pos} and ϵ_{torus_nor} are approximated based on the assumption that the osculating torus $T(s,t)$ is fitted to the mid-point of the surface patch in the leaf node. This assumption implies that the upper bound of the inequalities 6.11 and 6.12 have high chances to appear on the boundaries of the surface patch and in particular, one of the four corner points of the surface patch. To obtain ϵ_{torus_pos} and ϵ_{torus_nor}, we compute $|T(s,t) - S(u,v)|$ and $|N_t(s,t) - N(u,v)|$ for four corner points of the progenitor surface patch and pick the maximum values among them.

For ϵ_{offset_torus}, we note that the computed torus patches are osculated to the surface either on the outer equator of the torus ($T(0,0) = S(u_{mid}, v_{mid})$) or on the inner equator of the torus ($T(0,\pi) = S(u_{mid}, v_{mid})$), depending on the sign of the gaussian curvature $K = \kappa_1\kappa_2$ at $S(u_{mid}, v_{mid})$. That is, if two principal curvatures of the surface have the same sign, we fit the outer major circle to the surface and the inner major circle, otherwise. Depending on which side of torus we use to fit the surface, the bounding error values can be determined as follows.

1. If the outer circle is fit: We approximate ϵ_{offset_torus} by measuring the distance between $T_{offset}(s_{mid}, t_{mid})$ and the tetrahedron made up of four corner points of the offset surface (see Figure 6.6 (a)).

2. If the inner circle is fit: We find the distance between the tetrahedron made up of four corner points of the offset surface and four points in the mid-boundary of the offset torus patch. Then we take the maximum of four values as ϵ_{offset_torus} (see Figure 6.6 (b)).

In conclusion, we enclose the actual offset surface patch in the leaf node by a tetrahedron of which points are four corners of the actual geometry with



(a) (b)

Figure 6.6: Sample points to compute ϵ_{offset_torus} on (a) the outer torus and (b) the inner torus patches.

the bounding error of ϵ.

Bounding Error of a Non-leaf Offset Tetrahedron

Once the leaf bounding volumes are constructed, the bounding errors of all internal bounding volumes are computed from the bounding errors of their child nodes. Assume a non-leaf node n enclosing the surface $S(u, v)$ on $[u_1, u_2] \times [v_1, v_2]$ has four child nodes n_{11}, n_{12}, n_{21} and n_{22} enclosing $S(u, v)$ on $[u_1, u_{mid}] \times [v_1, v_{mid}]$, $[u_{mid}, u_2] \times [v_1, v_{mid}]$, $[u_1, u_{mid}] \times [v_{mid}, v_2]$ and $[u_{mid}, u_2] \times [v_{mid}, v_2]$ where $u_{mid} = \frac{u_1+u_2}{2}$ and $v_{mid} = \frac{v_1+v_2}{2}$. If the bounding error of each child node is ϵ_{ii} for $i \in \{1, 2\}$ and $j \in \{1, 2\}$, the bounding error ϵ_n of the node n

satisfies the following inequality:

$$\begin{aligned}\epsilon_n \leq & \max(\epsilon_{11}, \epsilon_{12}, \epsilon_{21}, \epsilon_{22}) \\ & + \max(dist(p_{11}, n), dist(p_{12}, n), dist(p_{21}, n), dist(p_{22}, n), dist(p_{center}, n))\end{aligned} \tag{6.16}$$

where $dist(p, n)$ is the minimum distance from a point p to the offset tetrahedron n and $p_{11}, p_{12}, p_{21}, p_{22}$ and p_{center} are $S(u_{mid}, v_1)$, $S(u_{mid}, v_2)$, $S(u_1, v_{mid})$, $S(u_2, v_{mid})$ and $S(u_{mid}, v_{mid})$, respectively. This inequality can be proven by the simple triangular inequalities constructed on the edges of the node n and n_{ij}s. The computation of finding the bounding errors of the offset tetrahedron with two children nodes is already shown by Kang et al. [42].

6.4.2 Finding Self-intersections in Offset Surface Using BVH

In this section, we explain how to use the BVH constructed in Section 6.4.1 to accelerate the detection of the self-intersections and redundancies in the offset surface. Suppose that the hierarchy of bounding volumes is already constructed for the surface $S(u, v)$. In general, the self-intersections of $S(u, v)$ are detected by testing the intersections between the bounding volumes, as shown in Algorithm 1. Here, a bounding volume is split into child nodes and tested whether they have intersections among them. Each child node is also tested to contain any self-intersections. The detection tests are executed recursively until the node becomes the leaf node.

Algorithm 1 Find self-intersection in BV n enclosing $S(u, v)$

1: **procedure** FINDSELFINTERSECTION(n)

2: **for** every child node n_i of n **do**

```
3:        for every child node $n_j$ of $n$ do
4:            if $n_i \neq n_j$ then
5:                FindIntersection($n_i, n_j$)
6:    for every child node $n_i$ of $n$ do
7:        FindSelfIntersection($n_i$)
8: procedure FindIntersection($n_i$, $n_j$)
9:    if $n_1$ is a leaf and $n_2$ is a leaf then
10:       if $n_1$ is adjacent to $n_2$ then
11:           return
12:   $\epsilon_1 \leftarrow$ bounding error of $n_1$
13:   $\epsilon_2 \leftarrow$ bounding error of $n_2$
14:   $d_{tetra} \leftarrow$ dist($tetrahedron_1$, $tetrahedron_2$)
15:   if $d_{tetra} \leq \epsilon_1 + \epsilon_2$ then
16:       if $n_1$ is a leaf then
17:           if $n_2$ is a leaf then
18:               return $(n_1, n_2)$
19:           else
20:               for every child node $n_i$ of $n_2$ do
21:                   FindIntersection($n_1, n_i$)
22:       else
23:           if $n_2$ is a leaf then
24:               for every child node $n_i$ of $n_1$ do
25:                   FindIntersection($n_i, n_2$)
26:           else
27:               for every child node $n_i$ of $n_1$ and $n_j$ of $n_2$ do
```

28: FindIntersection(n_i, n_j)

Algorithm 1 yields a list of pairs of leaf bounding volumes having the distance smaller than the sum of the bounding errors. Here, the self-intersections are detected only up to the resolution of the leaf node. Therefore, the additional step of identifying the intersection curve between the intersecting surfaces must be performed for the detected pairs of leaf nodes.

The key to the acceleration of Algorithm 1 is in the reduction of the number of comparisons between the bounding volumes. This can be either done by finding the non-relevant bounding volumes in the stages as early as possible and excluding them from further recursions, or by finding the geometric conditions to guarantee that two bounding volumes never overlap with each other. From now on, we explore the additional constraints to exclude non-relevant bounding volumes or bounding volume pairs that never contribute to the self-intersections of the trimmed offset surface.

Detection of the Normal Flipping in Offset Surface

In Section 5.3, we state that the region of the surface where the normal of the offset surface flips from the normal of the progenitor surface must be excluded and not included in the trimmed offset surface. The difference between the normal of the offset surface and the normal of the progenitor surface comes from Equation 5.10 and is reformulated as Equation 5.25. The geometric meaning of Equation 5.25 is that the offset surface flips its orientation when one of the principal curvatures of the surface at that point is larger than the inverse of the offset radius, but not both of the curvatures. This constraint is analogous to finding the local self-intersections or the fishtails of the planar

offset curves. By omitting the positive redundant terms from Equation 5.25, the function $f(u, v)$ that determines the normal flipping of the offset surface is simplified as follows:

$$f(u, v) = (\kappa_1 - \frac{1}{d})(\kappa_2 - \frac{1}{d}) \tag{6.17}$$

where κ_1 and κ_2 are the principal curvatures of the progenitor surface at $S(u, v)$ and d be the offset distance. If $f(u, v)$ is positive, the normals of both progenitor and offset surface are the same, and the offset surface faces reversely if $f(u, v)$ is negative. Therefore, the region in offset surface where $f(u, v)$ is non-positive must be excluded in the trimmed offset surface.

When the BVH is used to find the self-intersections of offset surface, the sign of $f(u, v)$ must be determined for each bounding volume: if $f(u, v)$ is non-positive for the entire domain of $O(u, v)$ in the bounding volume, the bounding volume and all of the child nodes of this volume no longer need to check self-intersections within them. If $f(u, v)$ is partially positive for the bounding volume of $O(u, v)$, the bounding volume must split, and the child nodes are tested again for the normal flipping. When the sign of $f(u, v)$ is tested for the bounding volume, we have to figure out either whether the upper bound of $f(u, v)$ is positive or whether the lower bound of $f(u, v)$ is negative for the domain of the bounding volume. In order to compute the exact upper or lower bounds of $f(u, v)$ for the specific domain representing the bounding volume, the curvature analysis tool such as that in the IRIT geometric modeling system might be useful. However, $f(u, v)$ involves the terms of the principal curvatures of the surface, and finding the maximum and minimum values of the principal curvatures on the continuous surface already requires a large number of computations.

Instead of the computation involving the curvature terms of the surface, we determine the sign of $f(u, v)$ of the bounding volume by comparing the value of $f(u, v)$ for some sample points on the offset surface. To this purpose, we determine the normal flipping on each leaf bounding volume by sampling the four corner points of the leaf node and test $f(u, v)$s only for the sampled points. First, if $f(u, v)$'s are negative for all of the corner points, we consider the normal of this leaf bounding volume as *fully flipped*. If all of the sample points have the positive $f(u, v)$ values, this bounding volume is marked as *not flipped*. Finally, if some of the four corner points have the positive $f(u, v)$ values whereas others have the negative $f(u, v)$, we mark the bounding volume as *partially flipped*, in which the offset surface flips somewhere inside the bounding volume. Only the nodes marked as *fully flipped* are excluded from the bounding volume intersection test in advance.

Figure 6.7 shows three different regions in the offset surface regarding normal flipping. The regions colored as grey in xyz-domain in Figure 6.7 (a) or white in uv-domain in Figure 6.7 (c) represent a collection of leaf nodes marked as *not flipped*, whereas the regions colored as green represent *partially flipped* leaf nodes. Leaf nodes marked as *fully flipped* are colored as orange in this figure. Figure 6.7 (b) and (d) are zoom-in views around the red circles in Figure 6.7 (a) and (c), respectively. Here the leaf nodes marked as *partially flipped* are colored as green, and the boundaries of the node are drawn as black lines. We can observe that *partially flipped* nodes successfully separates the *fully flipped* regions of the offset surface from *not flipped* regions despite the approximation we used to determine the signs of $f(u, v)$ of the bounding volumes.

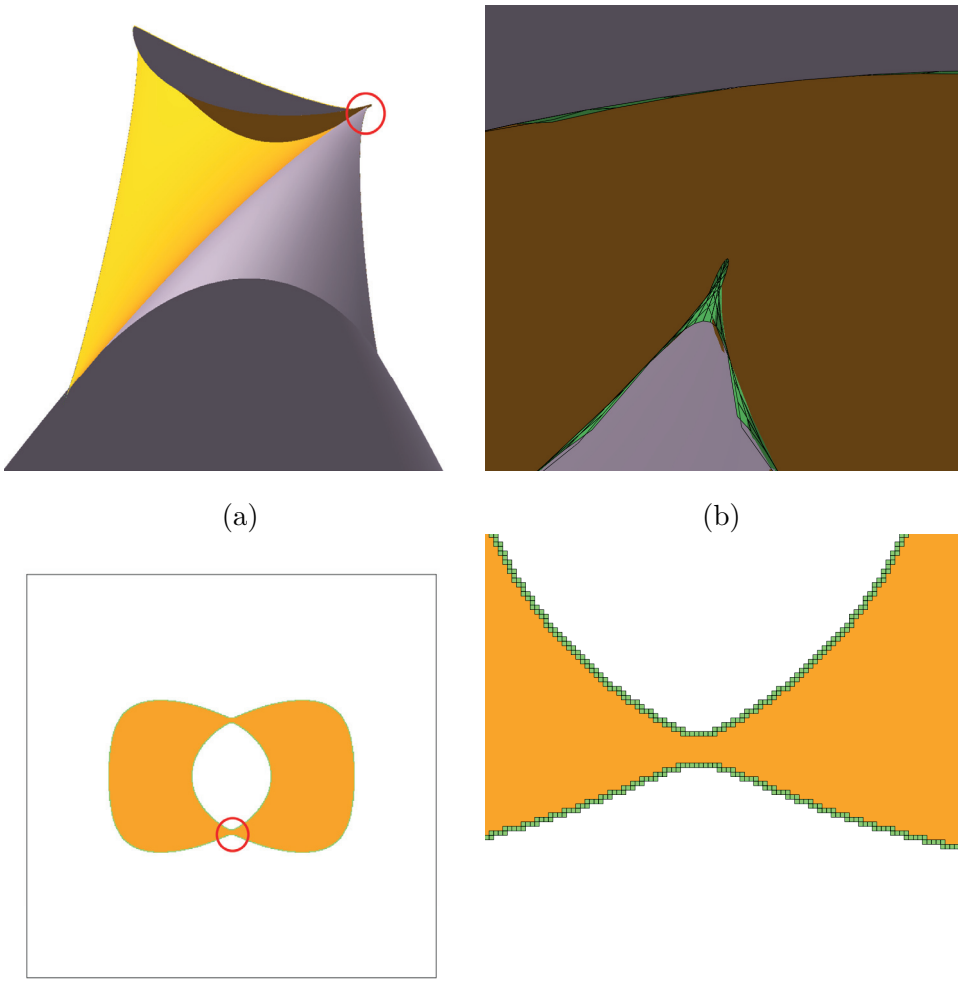


Figure 6.7: Normal-flipped regions of the offset surface (a) in the xyz-space and (b) the zoom-in view of (a); the corresponding region (c) in the uv-domain and (d) the zoom-in view of (c).

Projection Distance-based Offset Surface Trimming

The trimmed offset surface only contains the regions where the minimum distance to the progenitor surface is at least the offset distance. In other words,

if the minimum distance from some point (u_0, v_0) on the offset surface $O(u, v)$ to the progenitor surface $S(u, v)$ is smaller than offset distance, that point does not belong to the final trimmed offset region. For these points, we can find another point (u_1, v_1) on $S(u, v)$ such that $|O(u_0, v_0) - S(u_1, v_1)| < d$. In the case of planar offset curves, these points are expected to appear near the fishtails, where the offset curve abruptly changes its direction and orientation. In the offset surfaces, these points are expected to appear near the region where the offset surface flips its normal.

We can exclude a bounding volume from testing the self-intersections of the offset surfaces when there exists (u_1, v_1) such that $|O(u, v) - S(u_1, v_1)| < d|$ for the entire bounding volume. If we can find a sphere that has its center at some point on the progenitor surface $S(u_1, v_1)$ and the radius of the sphere is offset distance d, and this sphere contains the entire volume of the bounding volume, any bounding volume pairs containing this bounding volume no longer need to test intersections.

To simplify the computation, we use a more relaxed condition in the experiment to test whether each bounding volume can be trimmed based on the minimum distance to the progenitor surface. We measure the projection distance from the center of each offset tetrahedron (bounding volume) to the base surface. If the measured distance is less than the $\rho \cdot d$ where d is the offset distance, and ρ is some constant far larger than 1, we exclude any bounding volume pairs including this bounding volume from intersection tests. To find out this region, we measure the projection distance between the center of the offset tetrahedron and the base surface. If the measured distance is far less than the offset distance, we exclude the bounding volume from the self-intersection detection.

Self-intersection Tests of the Sibling Nodes

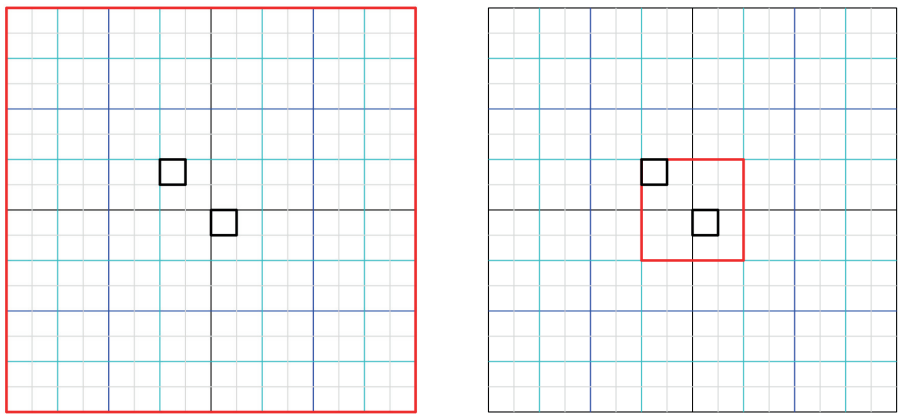
If we can find the additional constraints to ensure that the bounding volume does not have any local self-intersections in it, any pair of the bounding volumes under the branch of this bounding volumes in the BVH also have no intersections between them. Moreover, if two bounding volumes are determined not to intersect at all in the early stage of the intersection tests, we do not need to explore the sub-branches of these bounding volumes anymore, which can save the computational time spent on intersection tests. One constraint that ensures the local self-intersections on the surface is derived from the range of the normals of the surface. Suppose the normal of the surface be bounded by a normal cone $N(m, \alpha)$ where m is the axis of the cone, and α is the angle of the cone. If α is less than 90 degrees, there exist no loops in the intersection curves that generate the local self-intersections within the surface [77]. In the middle of finding the intersection between two bounding volume in Algorithm 1, therefore, if the parent node that contains two bounding volumes has the angle of the normal cone less than 90 degrees, it is guaranteed that the two bounding volumes never intersect. Therefore, for all of the bounding volume pairs encountered in the intersection tests, we first find the common parent node enclosing both bounding volumes. If the angle of the normal cone of the parent node is less than 90 degrees, two bounding volumes do not need to be subdivided to test intersection anymore.

Figure 6.8 shows an example of two bounding volumes and their common parent node shown in the uv-domain. The depth of the BVH in Figure 6.8 is 4, meaning that the surface is subdivided in u- and v-directions for four times to reach the leaf nodes. As we always subdivide the surface in both directions,

the domain of every bounding volume is a square, and the bounding volumes of depth h are aligned in the grid that has the interval of $\frac{1}{2^h}$ in both directions. In Figure 6.8, two nodes to test intersections are marked in thick black lines. The common bounding volume enclosing both nodes is the bounding volume of depth 0, which is marked as a red square in Figure 6.8 (a). This example demonstrates that even the small bounding volumes in proximity can have a large common parent node when the bounding volumes are placed slightly off from the grid structure of the BVH. Unfortunately, these situations are encountered frequently in the self-intersection tests of the offset surfaces because the bounding volumes of the offset surfaces have relatively large bounding errors compared to those of the progenitor surfaces. In extreme cases, almost all leaf nodes of the offset surface BVH interfere with the neighboring nodes. The smallest square-shaped node enclosing both bounding volumes in Figure 6.8 is the node of size 4, which has the same size as nodes of depth 2, as shown in Figure 6.8 (b). However, this node does not belong to the current BVH because it is shifted from the actual nodes of the same size in the BVH.

To remedy this problem, we add the auxiliary nodes to the BVH. The concept of constructing the auxiliary nodes that have overlaps in the parameter domain of the surface with the existing bounding volumes is similar to that of a *loose octree* where space is partitioned with some overlaps allowed [84]. The number of the nodes in the BVH increases approximately four times by shifting the nodes in u-, v- and diagonal directions in the parameter domain. We shift each node by half of the size of the node so that the new node always overlaps with the two nodes in the original BVH.

When the normal cone of the common parent node is checked for intersection, we only care for the parent node of which normal does not flip. When



(a) (b)

Figure 6.8: The common parent nodes (in red) of two leaf bounding volumes (in black) in (a) the originial bvh and (b) the bvh with the additional overlapping nodes.

not normal-flipped, the normal cone of the offset surface bounding volume is the same as the normal cone of the progenitor surface bounding volume. Thus, we compute the normal cone of the progenitor surface instead of that of the offset surface.

Algorithm 2 Find self-intersection in BV n enclosing $O(u, v)$

1: **procedure** FindSelfIntersection_OffsetSurface(n)
2: **for** every child node n_i of n **do**
3: **for** every child node n_j of n **do**
4: **if** $n_i \neq n_j$ **then**
5: FindIntersection_OffsetSurface(n_i, n_j)
6: $N(m, \alpha) \leftarrow$ normal cone of $S(u, v)$ corresponding to $O(u, v)$

```
    if n is not flipped and α < π/2 then
        return
    for every child node n_i of n do
        FindSelfIntersection_OffsetSurface(n_i)
procedure FindIntersection_OffsetSurface(n_i, n_j)
    if n_1 is flipped or n_2 is flipped then
        return
    if n_1 is a leaf and n_2 is a leaf then
        if n_1 is adjacent to n_2 then
            return
     cp ← common parent of n_i and n_j
    N(m, α) ← normal cone of S(u, v) corresponding to cp
    if α < π/2 then
        return
    ε_1 ← bounding error of n_1
    ε_2 ← bounding error of n_2
    d_tetra ← dist(tetrahedron_1, tetrahedron_2)
    if d_tetra ≤ ε_1 + ε_2 then
        if n_1 is a leaf then
            if n_2 is a leaf then
                return (n_1, n_2)
            else
                for every child node n_i of n_2 do
                    FindIntersection_OffsetSurface(n_1, n_i)
        else
```

31: **if** n_2 is a leaf **then**

32: **for** every child node n_i of n_1 **do**

33: FindIntersection_OffsetSurface(n_i, n_2)

34: **else**

35: **for** every child node n_i of n_1 and n_j of n_2 **do**

36: FindIntersection_OffsetSurface(n_i, n_j)

Algorithm 2 is the modified version of Algorithm 1 that includes the acceleration techniques discussed above. In particular, the modifications added in Algorithm 2 reduce the number of recursions in line 23-36 of the procedure **FindIntersection_OffsetSurface**.

Collision Detection within a Leaf Node

In experiments, we realize that further accelerations are desirable in practice. We apply the sub leaf level of self-intersection tests to trim more local self-intersections from the results. For the detected colliding bounding volume pairs, we sample the points on the subdivided surfaces and compute the convex hull of the sampled points. If the convex hulls of two bounding volumes do not overlap within the tolerance, we determine that two bounding volumes doe not intersect and eliminate the pair them from the colliding pair list.

6.4.3 Tracing Self-intersection Curves

The self-intersection detection algorithm proposed in the previous section yields a list of the leaf bounding volume pairs that have the potential intersections on the offset surface. For the detected pairs, we identify how the actual intersection curves proceed in the bounding volumes. The intersection curves

are computed first by finding the intersection points between the boundaries of the surface in one bounding volume and the surface in the other bounding volumes using the curve-surface-intersection algorithm. These intersection points serve as starting points to trace the intersection curves within the bounding volumes. We trace the intersection curves using a numerical tracing method, such as that of Wang [88]. Equation (6.18)–(6.20) represents the constraint equations used in the numerical tracing of the intersection curves.

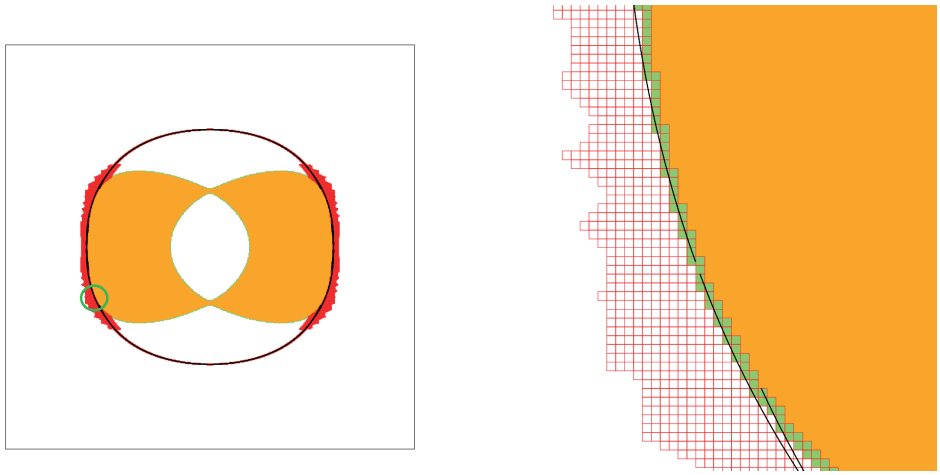
$$(S(u,v) - S(s,t)) \cdot d(N(u,v) + N(s,t)) = 0 \tag{6.18}$$

$$(S(u,v) - S(s,t) + dN(u,v)) \cdot S_s(s,t) = 0 \tag{6.19}$$

$$(S(u,v) - S(s,t) + dN(u,v)) \cdot S_t(s,t) = 0 \tag{6.20}$$

Equation (6.18) is derived from the equidistance constraint where the points on the offset self-intersection curves have the same distance d from the progenitor surface, whereas Equation (6.19) and (6.20) represent the projection constraints such that the points on the offset self-intersection curves must be orthogonal to the progenitor surface. We use the fourth-order Runge-Kutta method to trace these equations and yield a list of piecewise polylines as the intersection curves.

Figure 6.9 demonstrates the curve tracing results in the uv-domain. In Figure 6.9 (a), the intersection curves are only traced within the leaf nodes marked as self-colliding. When the tracing points reach to one boundary of the bounding volumes in the pair, we find the next colliding pair and continue the tracing. Figure 6.9 (b) shows an zoom-in view of the green circle marked in Figure 6.9 (a). We stop the numerical tracing around the singular point of $O(u,v)$ where $O_u(u,v) \times O_v(u,v)$ almost vanishes because the differentials



(a) (b)

Figure 6.9: (a) Intersection curves from numerical tracing. (b) The zoom-in view of the green circle marked on (a).

used in the numerical tracing becomes unstable around the singularities of $O(u, v)$. Around the singularities, we switch from the numerical tracing to the self-intersection computation using torus patches, just as mentioned in Section 5.3.

6.5 Experimental Results

We also have implemented the BVH-based offset surface trimming algorithm in C++, and measure the performance of the algorithm on an Intel Core i7-6700K 4.GHz PC with a 32GB main memory. We have applied the new trimming algorithm to the offset surface examples experimented in Chapter 5 to compare the execution time of both algorithms. In all of the experiments, we set the resolution of the leaf node to $\frac{1}{512}$ in both u and v directions, which

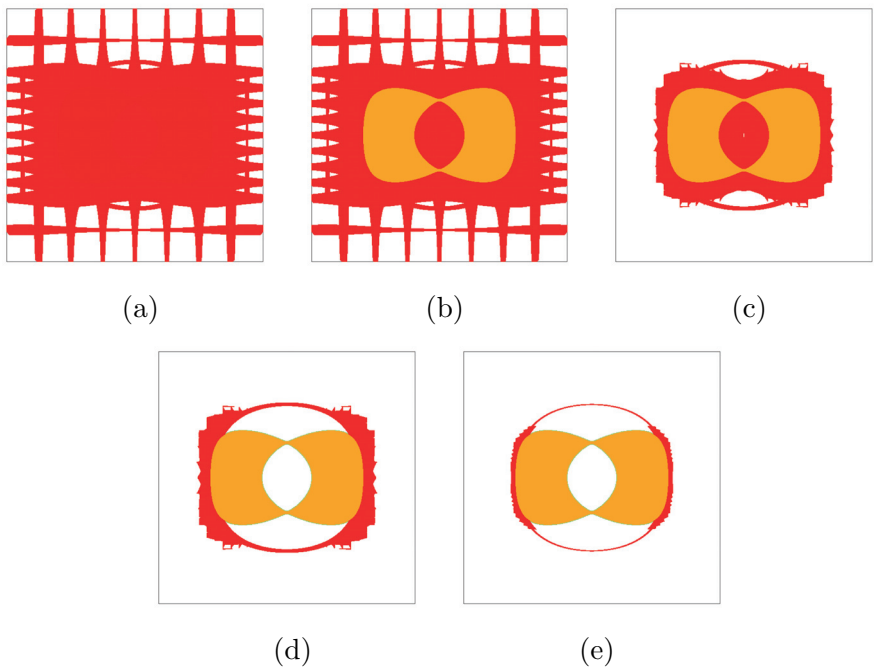


Figure 6.10: The colliding bounding volumes (in red) in the uv-domain as a result of acceleration techniques.

means that each offset BVH has $512 \times 512 = 262,144$ leaf nodes and their non-leaf nodes in the upper level of the BVH.

We first demonstrate the performance gain obtained from the various acceleration techniques proposed in Section 6.4.2. We compare the performance of Algorithm 1 in which the BVH is searched in a straightforward way to find the colliding bounding volume pairs, with Algorithm 2 in which the various pruning techniques are employed in the middle of the collision detection. Table 6.3 shows the number of bounding volume pairs collected when the different levels of acceleration techniques are applied. The first row in Table 6.3 shows the number of the pairs detected only with Algorithm 1. Considering the total

Example	**Srf10_d100**	**Srf10.1_d100**	**Srf10.5_d100**	**Srf12_d100**	**Srf15_d100**	**Srf20_d100**
(a) bvh test	96,044,558	99,303,997	94,158,738	100,840,316	49,300,378	34,191,068
(b) (a) + NF	22,451,691	25,883,557	24,535,419	28,095,774	11,251,693	6,194,687
(c) (b) + CP	14,408,327	16,692,112	15,593,383	18,141,306	6,059,363	3,489,823
(d) (c) + PD	10,081,595	11,859,908	10,626,848	6,883,775	4,398,072	2,474,107
(e) (d) + SL	298,118	336,489	337,176	315,034	210,106	202,781

Table 6.3: The number of bounding volume pairs in collision.

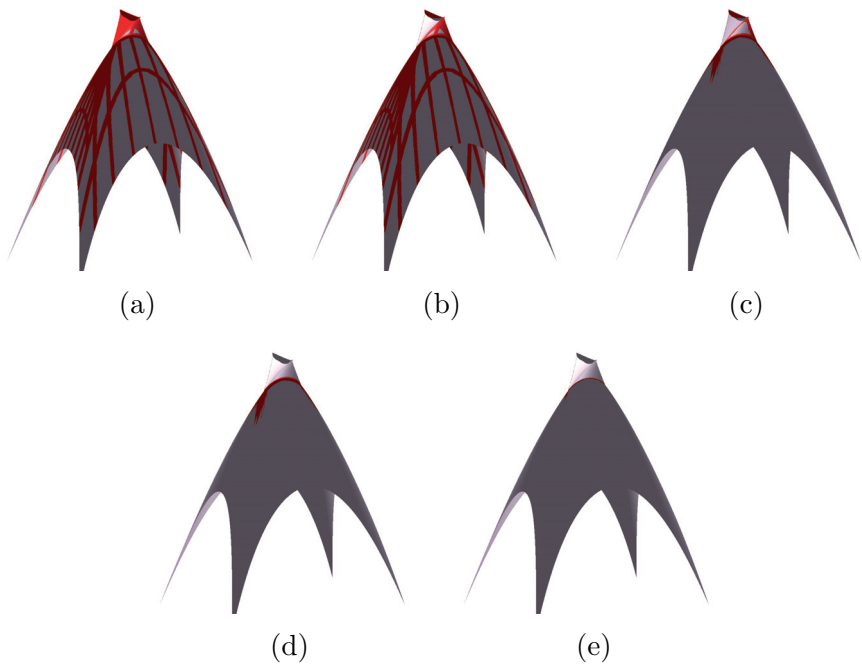


Figure 6.11: Collision pairs in xyz-domain as a result of acceleration techniques.

number of the leaf nodes is about 260K, the number of bounding volume pairs shown in the first row is relatively large: for instance, each leaf bounding volume in **Srf10_d100** collides with approximately 366 leaf bounding volumes in average. This is because of the bounding errors from the normal terms scaled by the offset distance, which thicken the bounding volume in the offset BVH and cause overlaps between the nearby bounding volumes. Fortunately, these numbers drastically decrease according to the addition of bounding volume filtering techniques. We determine the order of applying the trimming techniques based on the prior experiments to maximize the performance gains of the algorithm.

Figure 6.10 and 6.11 show the leaf bounding volumes of collision in the uv-domain and in the xyz-space. In Figure 6.10 (a), the detected bounding volumes are distributed in the entire offset surface because the redundant local self-intersections occur because of the thickness of offset surface bounding volumes. When the normal flipping regions are trimmed as in Figure 6.10 (b), the bounding volume pairs in the normal flipping region (colored as orange in Figure 6.10) of offset surface are trimmed away. In Figure 6.10 (c), the bounding volume pairs, each of which shares the common parent bounding volume, are eliminated. In this example, most of the bounding volume pairs in the flat or convex regions of the offset surface are pruned in this step. The bounding volumes closer to the progenitor surface than the offset distance d are eliminated in Figure 6.10 (d) and only the bounding volumes near the actual self-intersection curves are left after the sub leaf level of refinements as shown in Figure 6.10 (e).

Table 6.4 lists the execution time of the BVH-based surface offset trimming algorithm. We observe significant improvements in terms of the execution time: the self-intersection computation is accelerated up to approximately 500 times for several examples in the experiments. Note that, however, **Maekawa_1** only shows approximately $\times 2$ performance improvement. When the offset distance is relatively large compared with the problem size as shown in this example, the thickness of the offset surface bounding volume also increases cannot expect the drastic reduction of the number of colliding pairs.

Example	Description	Eq. Solving	Bvh
Srf12_d80	$x : y = 12 : 10$, $d = 0.12$	00:24:02	00:00:46
Srf12_d97	$x : y = 12 : 10$, $d = 0.1455$	00:39:14	00:01:19
Srf12_d99	$x : y = 12 : 10$, $d = 0.1485$	00:41:36	00:01:18
Srf12_d101	$x : y = 12 : 10$, $d = 0.1515$	00:44:04	00:01:20
Srf12_d103	$x : y = 12 : 10$, $d = 0.1545$	00:45:45	00:01:18
Srf12_d105	$x : y = 12 : 10$, $d = 0.1575$	00:47:54	00:01:26
Srf10_d100	$x : y = 10 : 10$, $d = 0.15$	01:10:31	00:01:20
Srf10.1_d100	$x : y = 10.1 : 10$, $d = 0.15$	01:09:22	00:01:27
Srf10.5_d100	$x : y = 10.5 : 10$, $d = 0.15$	00:59:09	00:01:20
Srf12_d100	$x : y = 12 : 10$, $d = 0.15$	00:41:11	00:01:23
Srf15_d100	$x : y = 15 : 10$, $d = 0.15$	00:30:50	00:00:49
Srf20_d100	$x : y = 20 : 10$, $d = 0.15$	00:27:15	00:00:45
Maekawa_1	Maekawa's example 1	00:50:04	00:26:29
Maekawa_2	Maekawa's example 2	00:20:12	00:00:06
Maekawa_3	Maekawa's example 3	20:08:16	00:00:24
Pipe	Seong's example	01:07:36	00:00:24
SweepCurve	Seong's example	00:42:34	00:00:19
Helix	Seong's example	01:24:11	00:00:25

Table 6.4: Execution time (in hh:mm:ss) of trimming algorithms using the multivariate equation solving (third column) and the BVH-baed acceleration techniques (in fourth column).

6.6 Summary

We have presented a new approach to trimming the offset curves and surfaces that accelerates the trimming algorithms presented in Chapter 4 and 5. The modified trimming algorithms capture the accurate topological behaviors of the self-intersections of the offset curves and surfaces while showing a significant performance improvement. In particular, the self-intersections in the near-singularities and the branching points are still revealed in the offset surface self-intersections. To accelerate the trimming algorithm, we construct the bounding volume hierarchy of the offset curves and surfaces from the position and the normal of the progenitor curves and surfaces. We also present various geometric conditions to prune the offset curve and surface BVHs. The experiment results show that the modified offset surface trimming algorithm accelerates the trimming $\times 500$ in the best case.

Chapter 7

Application of Trimmed Offset Surfaces: 3D Voronoi Diagram

7.1 Background

A *Voronoi diagram* is a partition that subdivides space into cells, and each cell contains all of the loci that are closer to the particular site than other sites in the space. In the classical definition, the sites of the Voronoi diagram consist of points in the plane. Here Voronoi cells are convex polygons constructed from intersecting half-planes between the sites. An edge of each Voronoi cell is a bisector line that is equidistant from two different sites, and a vertex is a point where there exist three equidistant sites among the sites. This standard Voronoi diagram has been well-studied and thoroughly analyzed in computational geometry [5]. Generalized Voronoi diagrams, on the other hand, have various shapes of sites from simple points, line segments [48], and curves [2] to more complex high-dimensional geometries such as spheres and cylinders [32],

and freeform surfaces or a combination of different geometries. If the sites are general objects, however, the construction of the Voronoi diagram becomes more intricated: Voronoi cells are split not only between distinct sites but also by the same sites.

Finding a Voronoi diagram of a planar curve is a dual problem of computing trimmed offset curves as a Voronoi diagram derives trimmed offsets and vice versa (Held [34]). Figure 7.1 illustrates the connection between a Voronoi diagram and trimmed offset curves of a planar curve. The boundaries of Voronoi cells are parts of the bisector lines that are equidistant to two different points on the curve. A point on the bisector line, which is distance d far away from the two sites on the progenitor curve, is also one of the self-intersection points of the offset curve with the offset distance d. A borderline separating two Voronoi cells is composed of a collection of self-intersection points of trimmed offset surfaces with varying offset distances. Finally, vertices of the Voronoi cells that have three equidistant sites are interpreted as points where two self-intersection points meet in trimmed offset curves.

The connection between a Voronoi diagram and trimmed offsets is also valid in 3D as well. Similar to that of planar curves, we define a Voronoi diagram of a freeform surface as a partition separating 3D space into a set of 3D cells in which each cell is closer to the particular surface patch than other patches of the surface. The 3D Voronoi diagram is also derived from the trimmed offset surfaces as follows. Given an offset surface $O(u, v)$ with the offset distance d, the self-intersection curves obtained from offset surface trimming are spatial curves that have the same distance d to at least two different sites on the progenitor surface. When the self-intersection curves of the trimmed offset surface of varying offset distances are collected, these curves

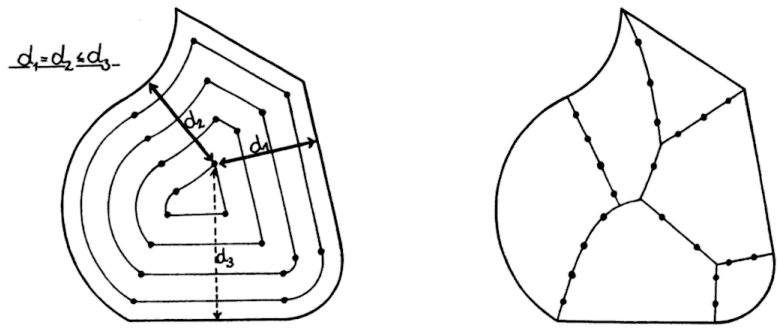


Figure 7.1: A relation between trimmed offset curves with varying offset distances (left) and a Voronoi diagram of a curve (right) (images from Held [35]).

will construct the bisector surfaces that become the boundaries of 3D Voronoi cells. Y-branch or X-branch points in the trimmed self-intersection curves imply that there exist more than three equidistant points on the progenitor surface, so the traces of those branch points form the spatial curve where two or more Voronoi cells meet.

Identifying the structure of the Voronoi diagram of a single closed curve or surface is also related to the problem of finding the medial axis transform (MAT) of the object. In CAGD, there have been a variety of methods to construct the medial axes for polyhedra, including, for instance, thinning, distance field computation, and surface sampling [32]. Nevertheless, few attentions are in the construction of the Voronoi diagram or medial axis transform of freeform surfaces.

In this chapter, we further investigate the relation between trimmed offset surfaces and a Voronoi diagram in 3D. Whereas there has been an extensive number of methods to construct Voronoi diagrams of planar curves [1, 2, 58],

to our best knowledge, there have no previous works on computing Voronoi diagrams of freeform surfaces from trimmed offset surfaces in CAGD yet. We identify the boundaries of Voronoi cells of a freeform surface by offsetting the surface with offset radius changing. Even though using discrete samples in offset distance, a collection of self-intersection curves and branching points in the self-intersection curves will reveal the structure of bisecting surfaces and trisecting curves, which are components of the Voronoi diagram of the progenitor surface.

In this chapter, we investigate the relationship between trimmed offset surfaces and voronoi digram in 3D. Whereas there already exists methods to compute trimmed offset curves from Voronoi diagram of planar curves, to our best knowledge, there has been previous work relating trimmed offset surfaces and Voronoi diagram of freeform surfaces in CAGD yet. We propose a method to identify Voronoi diagrams of freeform surfaces from trimmed offset surfaces with varying offset distances. The structure in self-intersection curves of the trimmed surface is interpreted in the terminology of Voronoi diagram and vice versa.

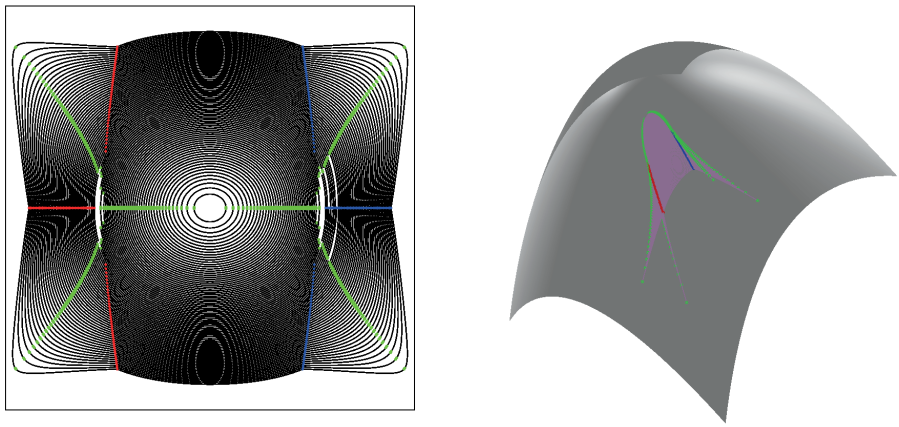
7.2 Approach

Given a progenitor surface $S(u, v)$, we compute the boundaries of Voronoi cells that appear on the concave side of the surface. These boundaries are coming from intersecting bisecting surfaces of a progenitor surface, which are equidistant to two distinct sites on the progenitor. Instead of intersecting bisectors, however, we derive the Voronoi diagram of the surface from a set of trimmed offset surfaces with offset radii continuously changing.

The idea of constructing the Voronoi cells from trimmed offsets is inspired by the offset surface trimming results shown in Section 5.4. In Figure 5.7, we show the evolution of self-intersection curves of trimmed offset surfaces when an offset radius changes from 0.12 to 0.1575. In this example, the self-intersection curve is a simple closed curve without any branching points when the offset radius is small (Figure 5.7 (i)), but starts to contain Y-branch points when an offset radius increases. Figure 5.7 (a) also indicates that locations of branching points are co-related between different trimmed offsets.

Examples in Section 5.4 include self-intersection curves for only a few instances of surface offset trimming. This is because the speed of the trimming algorithm proposed in Chapter 5 is not fast enough to compute multiple instances of trimmed offset surfaces. With the acceleration algorithm proposed in Chapter 6, we achieve a significant speed-up in computing self-intersection curves of trimmed offset surfaces, which enable us to compute hundreds of trimmed offset surfaces in a short time. Therefore, we compute the boundaries of the Voronoi diagram of the given freeform surface by densely sampling offset radii in the interval of interest and identifying the self-intersection curve structures of each offset surface.

Figure 7.2 shows the example of accumulated self-intersection curves of trimmed offsets in uv-domain and xyz-space. We represent branching points of self-intersection curves as red and blue dots and singular points using green dots. Each self-intersection curve (drawn as black in uv-domain and purple in xyz-space) is a d-isoline on a bisector surface of the progenitor surface. A curve that connects continuously evolving branch points is a tri-sector curve that occurs as a result of bisector surface intersection. The curve is also a trace of points that have three equidistant sites on the progenitor surface.



(a) (b)

Figure 7.2: Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space.

Figure 7.2 (b) shows that this example has five distinct bisector surfaces on the concave side, each of them separated by tri-sector curves. The boundaries of the bisector surfaces are either tri-sector curves that connect adjacent bisector surfaces or curves consisting of singular points in which the normals of trimmed offset surfaces vanish. By displaying the self-intersection curves of trimmed offsets altogether, we qualitatively identify the topological structure of the Voronoi diagram of the progenitor surface.

7.3 Experimental Results

We have constructed the boundaries of Voronoi cells for the freeform surfaces tested in Chapter 5 and 6. Bisector surfaces and tri-sector curves are derived from singular and branching points of self-intersection curves in trimmed offset

surfaces with varying offset radius. In experiments, we set offset radii tested in the previous chapters to the default offset radii. We then generate 100 samples of trimmed offset surfaces for the given progenitor surfaces, with intervals of 50 percent and 150 percent of the default offset radii. All of the self-intersection curves have been computed with the trimming algorithm proposed in Chapter 6, which accelerates the detection of self-intersections in offset surfaces by using the proposed bounding volume hierarchy.

Figure 7.3 shows Voronoi diagrams of two progenitor surfaces: the surface in the upper row comes from the fourth row in Figure 5.5 The example surface in the upper row is the progenitor surface from the fourth row in Figure 5.5, whereas the surface in the lower row is the progenitor surface from the fifth row in Figure 5.5. For both surfaces, we compute trimmed offset surfaces on the concave side of the progenitors, with offset radii varying from 0.075 to 0.225. The boundaries of Voronoi cells of both surfaces are composed of one broad main bisector surface and four side bisector surfaces separated from the main bisectors. The size of side bisector surfaces, however, is different between two surfaces: the upper surface that is more square-shaped has larger side bisector surfaces, meaning that the branch points of self-intersection curves start to develop from smaller offset radii.

When the progenitor surface is perfectly square-shaped as shown in Figure 7.4, there is no longer a main bisecting surface such as those in Figure 7.3, but only four side bisecting surfaces appear in the Voronoi diagram. Four singular points and four branching points converge to a single point in the middle in uv-domain when the offset radius decrease. In xyz-space, four distinct singular points collapse to the X-branch point in the self-intersection curve, and the apex of the bisecting surfaces is a sharp cuspidal point in xyz-space.

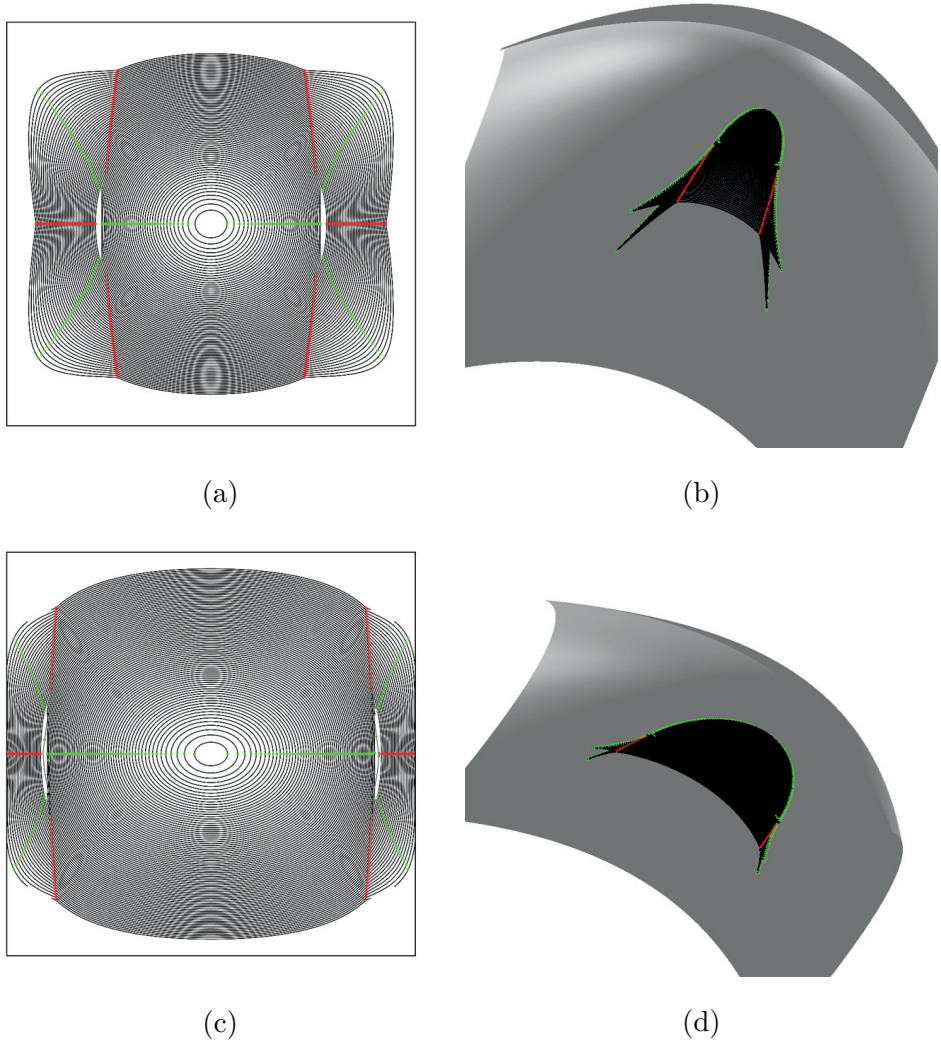
We also compute the Voronoi diagram structure for a more complicated B-spline surface. In Figure 7.5, we compute a collection of trimmed offset surfaces and their self-intersection curves for the pipe progenitor surface experimented in Figure 5.8 (see the left image of Figure 5.8). Black lines in Figure 7.5 represent the self-intersection lines of trimmed offset surfaces, whereas red lines represent traces of Y-branching points of trimmed offset surfaces. We also select three instances of trimmed offset surfaces and draw them in Figure 7.6. The corresponding self-intersection curves are drawn as blue curves in uv-domain (the above images) and also in xyz-domain with the offset surfaces. When the offset radius is small, the self-intersection curves of the trimmed offset surface consist of three loops in uv-domain, two of which appear in the same position in xyz-space. These loops start to grow when the offset radius increases and are merged to a single loop shown in the right images of Figure 7.6. Furthermore, when the offset radius grows, the new bisector surface starts to develop on the outer area of the pipe surface (see the isolated half-circles appeared in the upper right region in uv-domain). As demonstrated above, the geometric structure of the Voronoi diagram boundaries becomes complicated, even for such a simple pipe surface.

7.4 Summary

In this chapter, we construct the boundary elements of a Voronoi diagram of a freeform surface from the self-intersection curves of trimmed offset surfaces. Even though discretely sampled in offset radii, the development of self-intersection curves in shrinking or expanding offset surfaces reveals the geometrical structure of bisector surfaces and the connection of adjacent bisecting

surfaces on the boundary of Voronoi cells.

In the experimental results demonstrated in this chapter, we have some of the self-intersection curves around near-singular regions left disconnected because of the numerical instability in tracing self-intersection curves. For instance, Y-branching points and singular points must be merged into a single point in uv-domain in Figure 7.2. From the experimental results, though, we can easily observe that there exists some level of smoothness and continuity in the boundaries of bisecting surfaces. We hypothesize that self-intersection curves and bisecting surfaces missing near singularities be handled using this smoothness and continuity conditions but leave the topic as future work. Just as trimmed offset curves of a planar curve is derived from a Voronoi diagram of the curve and vice versa, the experiment results shown in this chapter implies that the construction of the Voronoi diagram of a freeform surface and the computation of trimmed offset surfaces are mutually beneficial to each other. Therefore, the Voronoi diagram of a freeform surface will also help to handle singularities in the self-intersection curves of offset surfaces.



(a) (b)

(c) (d)

Figure 7.3: Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space.

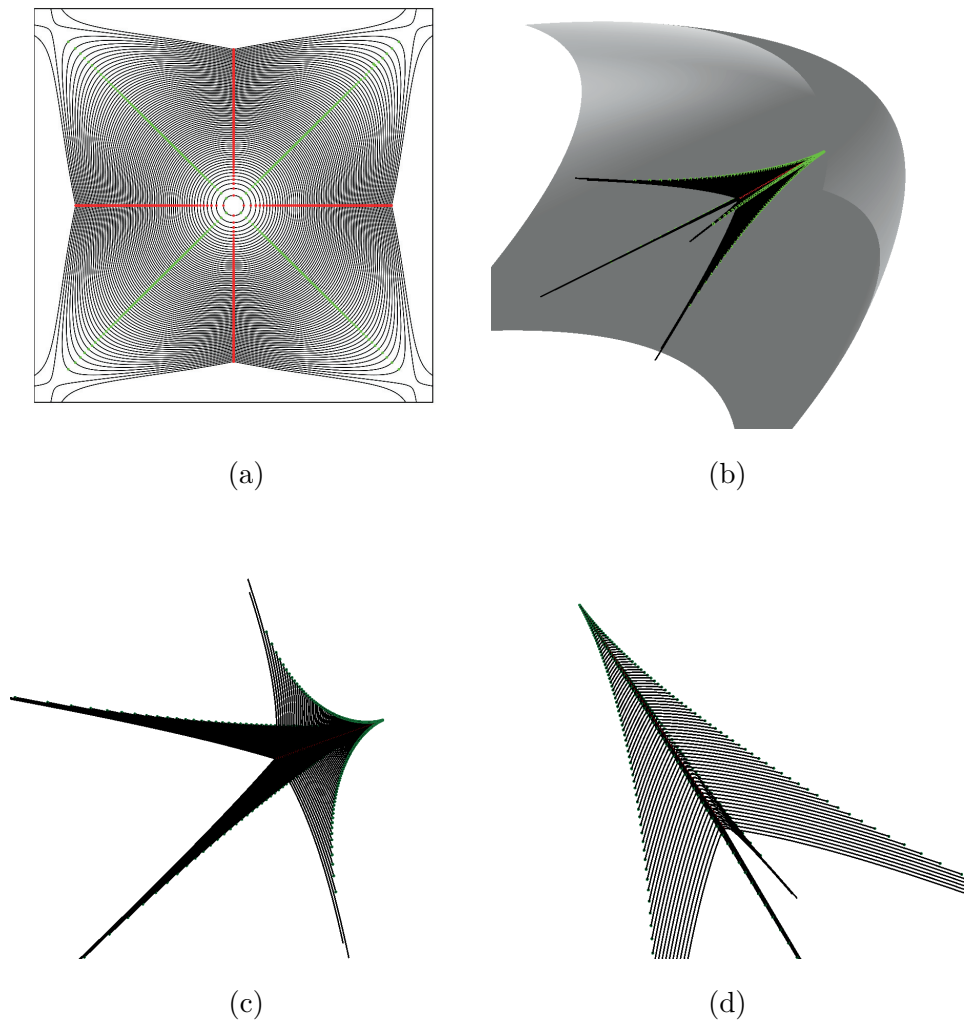
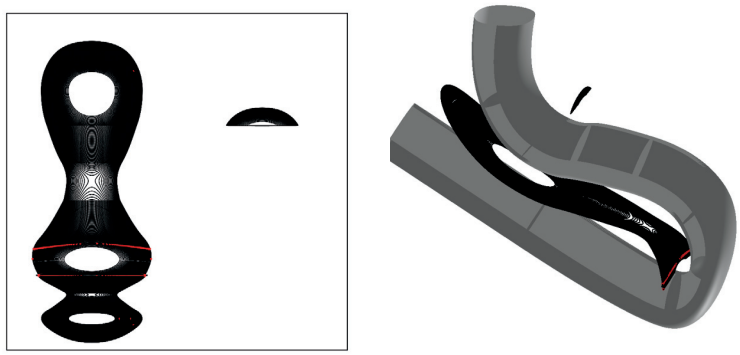


Figure 7.4: Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space.



(a) (b)

Figure 7.5: Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space.

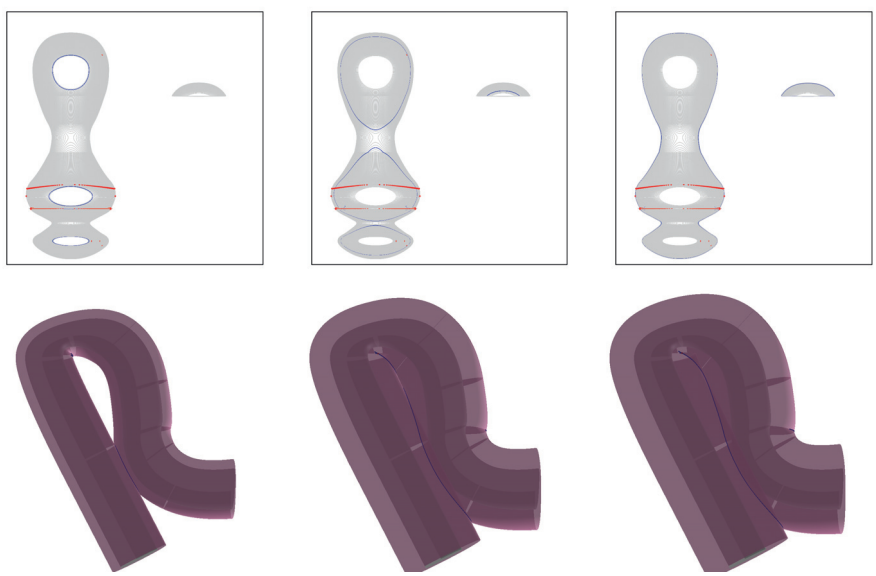


Figure 7.6: Self-intersection curves of varying offset distances in (a)uv-domain and (b) xyz-space.

Chapter 8

Conclusion

In this thesis, we have presented a geometric framework to detect the self-intersections and trim the redundant regions from the offset curves and surfaces. First, we identify the geometric structure of the self-intersections on the trimmed offset curves and surfaces. From the isosceles relation between the points of the progenitor curves or surfaces and their corresponding intersection points on the offset curves or surfaces, we construct the geometric constraint equations in the parameter domain of the progenitor curves or surfaces. The constraint equations of the self-intersections are formulated in a 2-dimensional parametric space in case of the offset curves, and in 4-dimensions in case of the offset surfaces. Additional inequality constraints are introduced to trim the redundant regions from offset curves and surfaces. These offset trimming algorithms enable us to capture the generic nature of the self-intersections of the offset curves and surfaces, including the self-intersections around near-singularities and the terminal points of the self-intersection curves that make

a branching structure.

We also present a method to accelerate the trimming algorithm of the offset curve and surface self-intersections, based on the observation that aforementiond equation solving has limitations on the performance of the algorithm. The modified trimming algorithm constructs the hierarchical structure of bounding volumes enclosing the offset curves or surfaces and accelerates the detection of the self-intersections with various pruning techniques applied to the bounding volume hierarchy.

Finally, we also investigate the relationship between the trimmed offset surfaces and the 3D Voronoi diagram of the progenitor surfaces. The systematic way of identifying the topology of self-intersection curves in the trimmed offset surfaces with the acceleration algorithm of trimming offset surfaces provides us hints for deriving the boundary elements of 3D Voronoi cells from the trimmed offset surfaces of varying offset distance d.

Trimming offset curves and surfaces is a complicated problem because of the structure of the self-intersections is complex, and algebraic approaches often derive the equations of high degree. The solution also generally involves pathological behaviors around singularities. We hope our trimming algorithm will be beneficial to identify the structure of the self-intersections of the offset curves and surfaces and improve the performance of the offset curve and surface trimming. Handling the singularities of the self-intersections of the offsets will also help us to identify the structures of the related constructs, such as that of the Voronoi diagram.

Bibliography

[1] O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Jüttler, and M. Rabl. Medial axis computation for planar free-form shapes. *Computer-Aided Design*, Vol. 41, No. 5, pp. 339–349, 2009.

[2] O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Jüttler, E. Pilgerstorfer, and M. Rabl. Divide-and-conquer for Voronoi diagrams revisited. *Computational Geometry*, Vol. 43, No. 8, pp. 688–699, 2010.

[3] S. Aomura and T. Uehara. Self-intersection of an offset surface. *Computer-Aided Design*, Vol. 22, No. 7, pp. 417–421, 1990.

[4] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi. Uniform coverage of automotive surface patches. *The Int'l J. of Robotics Research*, Vol. 24, No. 11, pp. 883–898, 2005.

[5] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys*, Vol. 23, No. 3, pp. 345–405, 1991.

[6] R. Barnhill, T. Frost, and S. Kersey. Self-intersections and offset surfaces. *Geometry Processing for Design and Manufacturing*, Robert Barnhill (Ed.), SIAM, 1992.

[7] G. Brunnett. Geometric modeling of parallel curves on surfaces. *Geometric Modelling. Computing*, Vol. 14, pp. 37–53, 2001.

[8] M. Campen and L. Kobbelt. Polygonal boundary evaluation of Minkowski sums and swept volumes. *Computer Graphics Forum*, Vol. 29, No. 5, pp. 1613–1622, 2010.

[9] J. Caravantes, G. M. Diaz-Toca, M. Fioravanti, L. Gonzalez-Vega, and I. Necular. An algebraic framework for computing the topology of offsets to rational curves. *Computer Aided Geometric Design*, Vol. 52-53, pp. 28–47, 2017.

[10] M. do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[11] J. Y. Chen and B. Ravani. Offset surface generation and contouring in computer-aided design. *Journal of Mechanisms, Transmissions and Automation in Design* , Vol. 109, No. 1, pp. 133–142, 1987.

[12] Y. Chen and C.L. Wang. Uniform offsetting of polygonal model based on Layered Depth-Normal Images. *Computer-Aided Design*, Vol. 43, No. 1, pp. 31–46, 2011.

[13] B. K. Choi and C. S. Jun. Ball-end cutter interference avoidance in NC machining of sculptured surfaces. *Computer-Aided Design*, Vol. 21, No. 6, pp. 371–378, 1989.

[14] E. S. Cobb. Design of sculptured surfaces using the B-spline representation. PhD dissertation, Computer Science Dept., The University of Utah, 1984.

[15] S. Coquillart. Computing offset of B-spline curves. *Computer-Aided Design*, Vol. 19, No. 6, pp. 305–309, 1987.

[16] G. Elber. *Free form surface analysis using a hybrid of symbolic and numerical computation*, PhD Thesis, Dept. of Computer Science, The University of Utah, 1992.

[17] G. Elber and E. Cohen. Error bounded variable distance offset operator for free form curves and surfaces. *Int'l J. of Computational Geometry & Applications*, Vol. 1, No. 1, pp. 67–78, 1991.

[18] G. Elber, I.-K. Lee, and M.-S. Kim. Comparing offset curve approximation methods. *IEEE Computer Graphics & Applications*, Vol. 17, No. 3, pp. 62–71, 1997.

[19] G. Elber and M.-S. Kim. The bisector surface of rational space curves. *ACM Trans. on Graphics*, Vol. 17, No. 1, pp. 32–49. 1998.

[20] G. Elber and M.-S. Kim. Computing rational bisectors. *IEEE Computer Graphics and Applications*, Vol. 19, No. 6, pp. 76–81. 1999.

[21] G. Elber and M.-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *Proc. of ACM symposium on Solid Modeling and Applications*, Ann Arbor, MI, June 4-8, 2001.

[22] G. Elber. Trimming local and global self-intersections in offset curves using distance maps. In *Proc. of the 10th IMA conference on the Mathematics of Surfaces*, Leeds, UK, pp. 213–222, September 2003.

[23] G. Farin. *Curves and surfaces for CAGD (fifth edition)*, Morgan Kaufmann, San Francisco, 2002.

[24] R. Farouki. The approximation of non-degenerate offset surfaces. *Computer Aided Geometric Design*, Vol. 3, No. 1, pp. 15–43, 1986.

[25] R. Farouki. *Pythagorean-Hodograph Curves*, Springer, Berlin, 2008.

[26] R. Farouki and C. Neff. Analytic properties of plane offset curves. *Computer Aided Geometric Design*, Vol. 7, No. 1–4, pp. 83–99, 1990.

[27] R. Farouki and C. Neff. Algebraic properties of plane offset curves. *Computer Aided Geometric Design*, Vol. 7, No. 1–4, pp. 101–127, 1990.

[28] R. Farouki. Pythagorean Hodograph curves: algebra and geometry inseparable. *Geometry and Computing*, Vol. 1, 2008.

[29] R. Farouki and J. Srinathu. A real-time CNC interpolator algorithm for trimming and filling planar offset curves. *Computer Aided Design*, Vol. 86, pp. 1-11, 2017.

[30] H.-Y. Feng and H. Li. Constant scallop-height tool path generation for three-axis sculptured surface machining. *Computer-Aided Design*, Vol. 34, No. 9, pp. 647–654, 2002.

[31] A. Gálvez, A. Iglesias, and J. Puig-Pey. Computing parallel curves on parametric surfaces. *Applied Mathematical Modelling*, Vol. 38, pp. 2398–2413, 2014.

[32] I. Hanniel, R. Muthuganapathy, G. Elber, and M. -S. Kim. Precise Voronoi cell extraction of free-form rational planar closed curves. *The Int'l J. of Computational Geometry and Applications*, Vol. 17, No. 5, pp. 453–496, 2007.

[33] W. Hansmann. Interactive design and geometric description of smooth transitions between curved surfaces. *Computers in Offshore and Arctic Engineering, Sixth International Symposium on Offshore Mechanics and Arctic Engineering*, Houston, Texas, ASME, New York, pp. 19–26, 1987.

[34] M. Held. Voronoi diagrams and offset curves of curvilinear polygons. *Computer-Aided Design*, Vol. 30, No. 4, pp. 287–300, 1998.

[35] M. Held. *On the computational geometry of pocket machining*, Springer, Berlin, 1991.

[36] V. D. Holla, K. G. Shastry, and B. G. Prakash. Offset of curves on tessellated surfaces. *Computer-Aided Design*, Vol. 35, No. 12, pp. 1099–1108, 2003.

[37] J. Hoschek. Spline approximation of offset curves. *Computer Aided Geometric Design*, Vol. 5, pp. 33–40, 1988.

[38] J. Hoschek and N. Wissel. Optimal approximate conversion of spline curves and spline approximation of offset curves. *Computer-Aided Design*, Vol. 20, No. 8, pp. 475–483, 1988.

[39] S.-M. Hu and J. Wallner. A second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design*, Vol. 22, No. 3, pp. 251–260, 2005.

[40] A. Iglesias, J. Gálvez, and J. Puig-Pey. Computational methods for geometric processing. Applications to Industry. *Computational Science – ICCS2001*, pp. 698–707, 2001.

[41] IRIT 10.0 User's Manual, Technion, 2009. http://www.cs.technion.ac.il/∼irit.

[42] Y. Kang, J. Jang, and M.-S. Kim. Deformable quad mesh for accelerated geometric operations. *Journal of the Korea Computer Graphics Society*, Vol. 2015, No. 7, pp. 73–74, 2015.

[43] M. S. Kim, E. J. Park, and S. B. Lim. Approximation of variable-radius offset curves and its application to Bézier brush-stroke design. *Computer-Aided Design*, Vol. 25, No. 11, pp. 684–698, 1993.

[44] T. Kim. Constant cusp height tool paths as geodesic parallels on an abstract Riemannian manifold. *Computer-Aided Design*, Vol. 39, pp. 477–489, 2007.

[45] Y.-J. Kim, Y.-T. Oh, S.-H. Yoon, M.-S. Kim, and G. Elber. Coons BVH for Freeform geometric models. *ACM Trans. Graph.*, Vol. 30, No. 6, pp. 169:1–169:8, 2011.

[46] 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*, Vol. 29, No. 7, pp. 555–564, 2012.

[47] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. In *Proc. of the National Academy of Sciences*, Vol. 95, pp. 8431–5, 1998.

[48] D. G. Kirkpatrick. Efficient computation of continuous skeletons. In *Proc. of the 20th Annual IEEE Symposium on FOCS*, pp. 18–27, 1979.

[49] R. Klass. An offset spline approximation for plane cubic splines. *Computer-Aided Design*, Vol 15, No. 5, pp. 297–299, 1983.

[50] R. Krasauskas and M. Peternell. Rational offset surfaces and their modeling applications. *Nonlinear Computational Geometry*, Vol. 151, pp. 109–135, 2010.

[51] C. Ravi Kumar, K. Shastry, and B. Prakash. Computing non-self-intersecting offsets of NURBS surfaces. *Computer-Aided Design*, Vol. 34, No. 3, pp. 209–228, 2002.

[52] R. Kunze, F.-E. Wolter, and T. Rausch. Geodesic Voronoi diagrams on parametric surfaces. In *Proc. Computer Graphics International 1997*, pp. 230–237, IEEE Press, Hasselt, Belgium, June 24–28, 1997.

[53] M.-H. Kyung, E. Sacks, and V. Milenkovic. Robust polyhedral Minkowski sums with GPU implementation. *Computer-Aided Design*, Vol. 67–68, pp. 48–57, 2015.

[54] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes Technical Report TR99-018, Dept. of Computer Science, UNC, 1999.

[55] E. Lee. Contour offset approach to spiral toolpath generation with constant scallop height. *Computer-Aided Design*, Vol. 35, No. 6, pp. 511–518, 2003.

[56] I.-K. Lee, M.-S. Kim, and G. Elber. Planar curve offset based on circle approximation. *Computer-Aided Design*, Vol. 28, No. 8, pp. 617–630, 1996.

[57] 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*, Vol. 58, No. 1, pp. 248–255, 2015.

[58] J. Lee, Y.-J. Kim, M.-S. Kim, and G. Elber. Efficient Voronoi diagram construction for planar freeform spiral curves. *Computer Aided Geometric Design*, Vol. 43, pp. 131–142, 2016.

[59] Y.-J. Liu. Semi-continuity of skeletons in two-manifold and discrete Voronoi approximation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 37, No. 9, pp. 1938–1944, 2015.

[60] 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*, vol. 26, no. 5, pp. 593–598, 2009.

[61] T. Maekawa and N .M. Patrikalakis. Computation of singularities and intersections of offsets of planar curves. *Computer Aided Geometric Design*, Vol. 10, pp. 407–429, 1993.

[62] T. Maekawa, W. Cho, and N. M. Patrikalakis. Computation of self-intersections of offsets of Bézier surface patches. *Journal of Mechanical Design: ASME Transactions*, Vol. 119, No. 2, pp. 275–283, 1997.

[63] T. Maekawa. An overview of offset curves and surfaces. *Computer-Aided Design*, Vol. 31, No. 3, pp. 165-173, 1999.

[64] W. Meng, S. Chen, Z. Shu, S. Xin, H. Fu, and C. Tu. Efficiently computing feature-aligned and high-quality polygonal offset surfaces. *Computers & Graphics*, Vol. 70, pp. 62–70, 2018.

[65] J. Mizrahi, S.-J. Kim, I. Hanniel, M.-S. Kim, and G. Elber. Minkowski sum computation of B-spline surfaces. *Graphical Models*, Vol. 91, pp. 30–38, 2017.

[66] N. M. Patrikalakis and L.Bardis, Offsets of curves on rational B-spline surfaces. *Engineering with Computers*, Vol. 5, No. 1, pp. 39–46, 1989.

[67] N. M. Patrikalakis and L.Bardis, Localization of rational B-spline surfaces. *Engineering with Computers* , Vol. 7, No. 4, pp. 237–252, 1991.

[68] N. M. Patrikalakis and T. Maekawa. *Shape interrogation for computer aided design and manufacturing*, Springer-Verlag, Heidelberg, 2002.

[69] D. Pavic and L. Kobbelt. High-resolution volumetric computation of offset surfaces with feature preservation. *Computer Graphics Forum*, Vol. 27, No. 2, pp. 165–174, 2008.

[70] D. Pekerman, G. Elber, and M.-S. Kim. Self-intersection detection and elimination in freeform curves and surfaces. *Computer-Aided Design*, Vol. 40, No. 2, pp. 150–159, 2008.

[71] B. Pham. Offset approximation of uniform B-splines. *Computer-Aided Design*, Vo. 20, No. 8, pp. 471–474, 1988.

[72] B. Pham. Offset curves and surfaces: a brief survey. *Computer-Aided Design*, Vol. 24, No. 4, pp. 223-229, 1992.

[73] T. Rausch, F.-E. Wolter, and O. Sniehotta. Computation of medial curves on surfaces. *The Mathematics of Surfaces VII*, pp. 43–68, 1997.

[74] J. R. Rossignac and A. A. G. Requicha. Offsetting operations in solid modelling. *Computer Aided Geometric Design*, Vol. 3, No. 2, pp. 129-148, 1986.

[75] G. Rousseau, R. Wehbe, J. Halbritter, and R. Harik. Automated fiber placement path planning: a state-of-the-art review. *Computer-Aided Design and Applications*, Vol. 16, No. 2, pp. 172–203, 2019.

[76] R. Sarma and D. Dutta. The geometry and generation of NC tool path. *Trans. of the ASME: J. of Mechanical Design*, Vol. 119, No. 2, pp. 253–258, 1997.

[77] T.W. Sederberg and R.J. Meyers. Loop detection in surface patch intersections. *Computer Aided Geometric Design*, Vol. 5, No. 2, pp. 161–171, 1988.

[78] J.-K. Seong, G. Elber, and M.-S. Kim. Trimming local and global self-Intersections in offset curves/surfaces using distance maps. *Computer-Aided Design*, Vol. 38, No. 3, pp. 183–193, 2006.

[79] Z. Sir, R. Feichtinger, and B. Jüttler: Approximating curves and their offsets using biarcs and Pythagorean hodograph quintics. *Computer-Aided Design*, Vol. 38, No. 6, pp. 608–618, 2006.

[80] M. Spivak. *A comprehensive introduction to differential geometry: volume three*, 2nd Ed., Publish or Perish, Inc., Houston, 1979.

[81] K. Sureh and D. C. H. Yang. Constant scallop-height machining of free-form surfaces. *J. of Engineering for Industry*, Vol. 116, No. 2, pp. 253–259, 1994.

[82] H.-Y. Tam, H.-W. Law, and H. Xu. A geometric approach to the offsetting of profiles on three-dimensional surfaces. *Computer-Aided Design*, Vol. 36, No. 10, pp. 887–902, 2004.

[83] W. Tiller and E. Hansen. Offsets of two dimensional profiles. *IEEE Computer Graphics and Applications*, Vol 4, No. 9, pp. 36–46, 1984.

[84] Loose Octrees. *Game Programming Gems*, Charles River Media, pp. 444–453, 2000.

[85] D.-E. Ulmet. Geodesic offsets of spline curves on spline surfaces: an industrial perspective. In *Proc. of 24th National Conf. of Geometry and Topology*, Timisoara, Romania, July 5–9, 1994, pp. 263–274.

[86] G. Varadhan and D. Manocha. Accurate Minkowski sum approximation of polyhedral models. *Graphical Models*, Vol. 68, No. 4, pp. 343–355, 2006.

[87] C.L. Wang and D. Manocha. GPU-based offset surface computation using point samples. *Computer-Aided Design*, Vol. 45, No. 2, pp. 321–330, 2013.

[88] Y. Wang. Intersection of offsets of parameteric surfaces. *Computer Aided Geometric Design*, Vol. 13, No. 5, pp. 453–465, 1996.

[89] F.-E. Wolter and S. Tuohy. Approximation of high-degree and procedural curves. *Engineering with Computers*, Vol. 8, No. 2, pp. 61–80, 1992.

[90] S.-Q. Xin, X. Ying, and Y. He. Efficiently computing geodesic offsets on triangle meshes by the extended XinWang algorithm. *Computer-Aided Design*, Vol. 43, No. 11, pp. 1468–1476, 2011.

[91] J. Xu, Y. Wang, X. Zhang, and S. Chang. Contour-parallel tool path generation for three-axis mesh surface machining based on one-step inverse forming. In *Proc. of MechE Part B: J. of Engineering Manufacture*, Vol. 227, No. 12, pp. 1800–1807, 2013.

[92] J. Xu, Y. Sun, and L. Zhang, A mapping-based approach to eliminating self-intersection of offset paths on mesh surfaces for CNC machining. *Computer-Aided Design* , Vol. 62, pp. 131–142, 2015.

초 록

오프셋 곡선 및 곡면은 computer-aided design (CAD)와 computer-aided manufacturing (CAM)에서 널리 이용되는 연산들 중 하나이다. 하지만 실용적인 활용을 위해서는 오프셋 곡선 및 곡면에서 생기는 자가 교차를 찾고 이를 기준으로 오프셋 곡선 및 곡면에서 원래의 곡선 및 곡면에 가까운 불필요한 영역을 제거하여야한다. 본 논문에서는 오프셋 곡선 및 곡면에서 생기는 자가 교차를 계산하고, 오프셋 곡선 및 곡면에서 생기는 불필요한 영역을 제거하는 알고리즘을 제안한다.

본 논문은 우선 오프셋 곡선 및 곡면의 자가 교차점들과 그 교차점들이 기인한 원래 곡선 및 곡면의 점들이 이루는 평면 이등변 삼각형 관계로부터 오프셋 곡선 및 곡면의 자가 교차점의 제약 조건을 만족시키는 방정식들을 세운다. 이 제약식들은 원래 곡선 및 곡면의 변수 공간에서 표현되며, 이 방정식들의 해는 다변수 방정식의 해를 구하는 solver를 이용하여 구한다. 오프셋 곡면의 경우, 원래 곡면의 주곡률 중 하나가 오프셋 반지름의 역수와 같을 때 오프셋 곡면의 법선이 정의가 되지 않는 특이점이 생기는데, 오프셋 곡면의 자가 교차 곡선이 이 부근을 지날 때는 자가 교차 곡선의 계산이 불안정해진다. 따라서 자가 교차 곡선이 오프셋 곡면의 특이점 부근을 지날 때는 오프셋 곡면을 접촉 토러스로 치환하여 더 안정된 방법으로 자가 교차 곡선을 구한다. 계산된 오프셋 곡면의 자가 교차 곡선으로부터 교차 곡선의 xyz-공간에서의 말단 점, 가지 구조 등을 밝힌다.

본 논문은 또한 바운딩 볼륨 기반의 오프셋 곡선 및 곡면의 자가 교차 곡선 검출을 가속화하는 방법을 제시한다. 바운딩 볼륨은 기저 곡선 및 곡면을 단순한 기하로 감싸고 기하 연산을 수행함으로써 가속화에 기여한다. 오프셋 곡면의 자가 교차 곡선을 구하기 위하여, 본 논문은 오프셋 곡면의 바운딩 볼륨 구조를

기저 곡면의 바운딩 볼륨과 기저 곡면의 법선 곡면의 바운딩 볼륨의 구조로부터 계산하며 이때 각 바운딩 볼륨의 두께를 계산한다. 또한, 바운딩 볼륨 중에서 실제 오프셋 곡선 및 곡면의 자가 교차에 기여하지 않는 부분을 깊은 재귀 전에 찾아서 제거하는 여러 조건들을 나열한다.

한편, 자가 교차가 제거된 오프셋 곡선 및 곡면은 기저 곡선 및 곡면의 보로노이 구조와 깊은 관련이 있는 것이 알려져 있다. 본 논문에서는 자유 곡면의 연속된 오프셋 곡면들로부터 자유 곡면의 보로노이 구조를 유추하는 방법을 제시한다. 특히, 오프셋 곡면의 자가 교차 곡선 상에서 나타나는 가지 점이나 말단 점과 같은 특이점들이 자유 곡면의 보로노이 구조에서 어떻게 해석되는지 제시한다.

주요어: 오프셋 곡선, 오프셋 곡면, 오프셋 자가 교차, 오프셋 특이점, 자가 교차 곡선 구조, 접촉 토러스, 바운딩 볼륨 구조, 자유곡면의 보로노이 다이어그램
학번: 2016-30283

감사의 글

먼저 4년 간의 박사 학위 과정 동안 부족한 저를 열의와 인내로 지도해주신 김명수 교수님께 감사를 드립니다. 교수님의 지도 덕분에 CAGD라는 분야를 접하고, 무사히 박사 과정을 끝마칠 수 있었습니다. 또한 예심에서 종심에 이르는 기간 동안 여러 조언들을 주셔서 이 논문의 완성에 도움을 주신 신영길 교수님, 이제희 교수님, 경민호 교수님, 윤승현 교수님께 깊은 감사를 드립니다. 또한, 이 논문에서 사용하였던 다변수 함수 솔버인 IRIT의 사용법 및 다변수 제약식 구성에 큰 도움을 주신 Technion의 Gershon Elber 교수님께도 감사를 드립니다.

한편, 지난 4년 동안 함께 랩 생활을 하며 여러 도움을 주었던 3차원 모델링 및 처리 연구실의 연구원 여러분들께도 감사를 드립니다. 이미 졸업한 윤구, 종인, 방달, 준오, 이번에 같이 졸업할 택희, 상준, 그리고 한동안 랩에 계속 남아 있을 영진, 상현, 민규, 유경 등, 서로 도움을 주고받으며 여러모로 더 깊은 배움을 얻을 수 있었던 4년이었습니다.

멀리 떨어져 살지만 한결같은 응원을 보내준 언니, 형부와 조카 준석에게도 감사를 드립니다. 마지막으로 긴 시간 동안 옆에서 노심초사하시며, 그럼에도 불구하고 항상 응원해주셨던 부모님께 깊은 감사드립니다.