Scalable Geometric Processing Techniques with Applications in Characterizing Additively Manufactured Composites

by

Xiang Li

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sara McMains, Chair
Professor Hayden Taylor
Professor Jonathan Shewchuk

Fall 2021

Scalable Geometric Processing Techniques with Applications in Characterizing Additively Manufactured Composites

Abstract

Scalable Geometric Processing Techniques with Applications in Characterizing Additively Manufactured Composites

by

Xiang Li

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Sara McMains, Chair

Fiber-reinforced polymer (FRP) composites are widely used in aerospace, marine, automotive, and other industries due to their superior strength-to-weight ratio and corrosion resistance. Analyzing FRP cross-sectional micrographs is one of the most widely used approaches for defect detection, quality inspection, failure analysis, and computational materials modeling. Although micrographs are widely used, a lack of specialized image and geometric processing techniques forces materials science researchers to manually analyze images, which makes the analysis process time-consuming and error-prone.

In this research, efficient and scalable geometric processing algorithms are proposed in order to characterize FRP materials and inspect their microstructure from microscope images. We develop two methods to automatically identify a major defect as well as microstructural feature in FRP composites: the resin-rich areas, which refer to areas of reduced strength caused by the lack of fiber reinforcements. We apply the concept of alpha-shapes and alpha-hulls to formalize mathematical definitions of the boundaries of resin-rich areas, and design efficient and scalable algorithms to compute the defined boundaries. In addition, a fiber recognition algorithm that automatically identifies and evaluates the breakage of the fiber cross-sections, and a GPU-based algorithm that efficiently constructs Voronoi diagrams of spheres/circles, are designed.

These methods enable us to provide statistical analyses to quantitatively characterize the identified resin-rich areas. The rigorous mathematical definition of resin-rich areas and ability to collect thorough statistics will facilitate better understanding and quantification of the relationship between resin-rich areas and material properties.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my gratitude and thanks to my advisor, Prof. Sara McMains, for providing me invaluable guidance and support throughout my graduate studies at Berkeley. With her consideration and patience, I have learned not only how to be a good researcher, but also how to be a good communicator in both teaching and writing. She also acted like a family member in my daily life, providing priceless suggestions when I met health or life problems. It is my honor and fortune to have Prof. McMains as my advisor.

I would like to thank my dissertation and qualifying exam committee members, Prof. Hayden Taylor and Prof. Jonathan Shewchuk, for their enlightening suggestions on my research and helpful feedback on my dissertation draft. Suggestions on considering convolution and morphological operations from Prof. Taylor and an anonymous paper reviewer inspired me in the development of what became my distance-transform-based algorithm. Prof. Shewchuk's computational geometry course and his expertise on Voronoi diagrams and Delaunay triangulation (and his Triangle library!) helped me better understand these geometric concepts and apply them in real-world applications.

I would like to thank Prof. Alice Agogino, who was the advisor of my master's study and a member of my qualifying exam committee. My study in her Berkeley Emergent Space Tensegrities Lab was delightful and memorable. I would also like to thank Prof. Grace Gu for being a member of my qualifying exam committee. I enjoyed Prof. Gu's composites class and gained a lot from her guidance on composite materials.

I would like to thank my lab colleagues Sara Shonkwiler, Hannah Budinoff, Youngwook Kwon, Bodi Yuan, and Jerenimo Mora from Prof. McMains' Computer-Aided Design and Manufacturing Lab, for their constructive comments and suggestions on my research. They are also my close friends in Berkeley; my life in Berkeley was made extremely interesting and rich with their friendship. I would like to especially thank Sara Shonkwiler for her "online" support with research discussions and manuscript revisions on multiple papers during the COVID period; and Hannah Budinoff for our collaboration (with Sara as well) on writing NSF grant proposals, which taught me a lot about proposal writing and planning.

I would like to thank Sushrut Pavanaskar for introducing me to Arevo Inc., which provided me the opportunity to engage in real-world manufacturing projects on 3D printing and composite materials. I would like to thank my supervisor, Peter Woytowitz, and my co-workers, Rick Fenrich, Juraj Vanek, Chih-Cheng Ho, Dantong He, Sohil Nandu, and Zhe Liu, for their continuous guidance and consideration at Arevo Inc. I would also like to thank Danning Zhang for providing the test images and for useful discussions about quality inspection of composite materials from microscope images.

I would like to thank my other co-authors, Adarsh Krishnamurthy and Iddo Hanniel, for their support and useful advice when I was a newbie in academic research.

Portions of this work were supported by National Science Foundation grant 1331352. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

I would like to thank Zhongyin Hu for inviting me to work in the Computer-Aided Design and Manufacturing Lab and collaborate on her Voronoi diagram project. Without her help, I may not have had the opportunity to be a PhD student.

I would like to thank my friends Qianqian Ma, Tiecheng Ma, Bing Guo, Qian Chen, and Haifeng Bao, for their care and encouragement over the past ten years. I would also like to thank Zi Wang and Wei Quan for accommodating me for years during my internship in Silicon Valley.

Finally, I would like to express my deepest gratitude to my family: my parents, my grandparents, and my fiancée, Linyi. I could not have done this without your constant love, belief, and support.

Chapter 1

Introduction

1.1 Research motivation

3D printing (3DP), or additive manufacturing, constructs 3D physical objects directly from digital models by successively depositing material layer by layer. The materials, methods, and applications of 3DP have been widely investigated due to advantages such as flexible customization, quick start to production, and the ability to fabricate complex geometries. Traditional 3DP systems use isotropic materials, such as metals or plastic filaments, whose mechanical properties are identical in all directions (though the layered prints are anisotropic). Recently, however, significant technical advances in 3DP processes have enabled printing with anisotropic materials to manufacture continuous fiber-reinforced polymer (FRP) composites [1, 2]. Compared to typical 3DP metal or plastic parts, FRP parts have superior strength-to-weight ratio in the fiber directions (e.g. lightweight 3D printed FRP parts have greater tensile strength than aluminum parts [2]).

Continuous FRP composites are anisotropic and heterogeneous materials that are made of two constituents: the fiber reinforcements, which are the primary load carrying portion of the composites; and the polymer resin matrix, which binds the fibers together, transfers and redistributes stresses among fibers, and protects fibers from aggressive (thermal and chemical) environments.

As reported in [3], the global market size of composite materials was estimated at 89.04 billion USD in 2019, and is expected to keep expanding at a 7.6% growth rate from 2020 to 2027. In the composite markets, FRP composites accounted for more than 95% of the market share.

Although FRP composites are increasingly popular in industry, microstructural characterization and defect detection for composite parts remains largely manual and inefficient. Defect detection, failure analysis, and material and mechanical properties of a composite part are often studied by characterizing its cross-sectional microscope images [4]. In continuous FRP 3D printing, parts are typically printed with a unidirectional or cross-ply laminate structure [5, 6], in which all fibers in the part, or in the same layer respectively, have the same

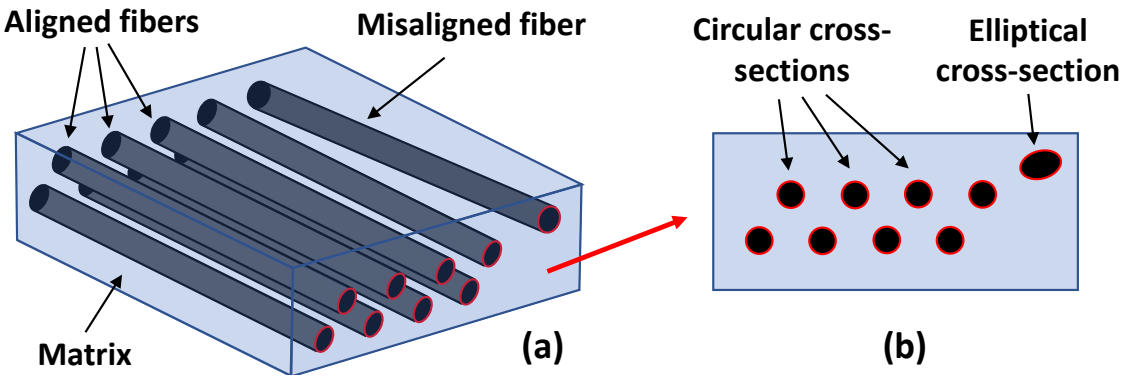

Figure 1.1: Diagram of (a) a unidirectional FRP composite; (b) its transverse cross-section with aligned and misaligned fiber cross-sections appearing as circles and ellipses respectively.

direction. Cross-sectional micrographs, taken transversely to the fiber direction, can effectively assist inspection and analysis of the material microstructure of each 3D printed layer (cross-ply structure) or the whole printed part (unidirectional structure). In FRP materials, ideally fibers are aligned and thus have circular cross-sections in the transverse microscope images, but misaligned fibers may exist, indicating areas of reduced strength; these appear as ellipses (Fig. 1.2).

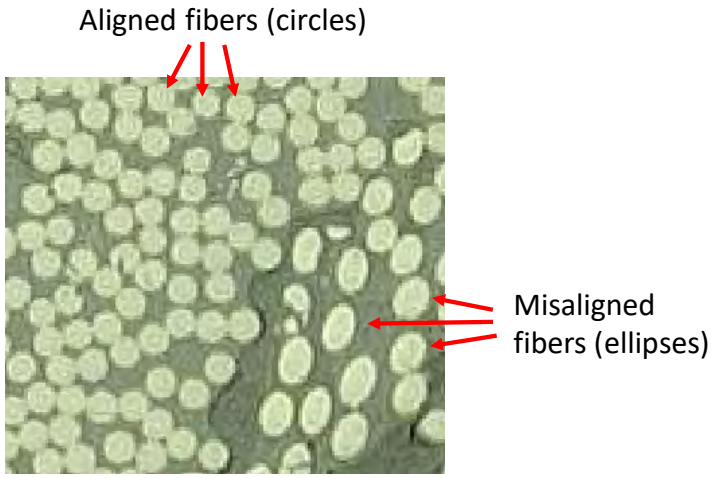

Figure 1.2: An example of aligned fibers (circular cross sections) and misaligned fibers (elliptical cross sections).

During development of FRP 3D printing processes, defect detection from microscope images is an important tool to help researchers analyze the printed parts and improve their

part quality. For FRP composites, common defects include voids, inclusions, and resin-rich areas [7]. Detecting voids and inclusions are both relatively well-studied in the material science community, and are now supported by image processing software such as ImageJ [8].

Compared to voids and inclusions, the detection of resin-rich areas is both more complex and less well-studied. Unevenly distributed fibers in the resin results in areas with insufficient reinforcement; such areas are commonly called *resin-rich areas* (Fig. 1.3) and lead to impaired mechanical properties and potential part failures [9, 10]. Currently, because of the lack of specialized image and geometric processing techniques, resin-rich area detection from microscope images is performed manually by experienced researchers [11], which makes the process time-consuming and error-prone. Fully automated, end-to-end analysis algorithms are required to efficiently and accurately process large amount of image data and characterize their microstructural properties.

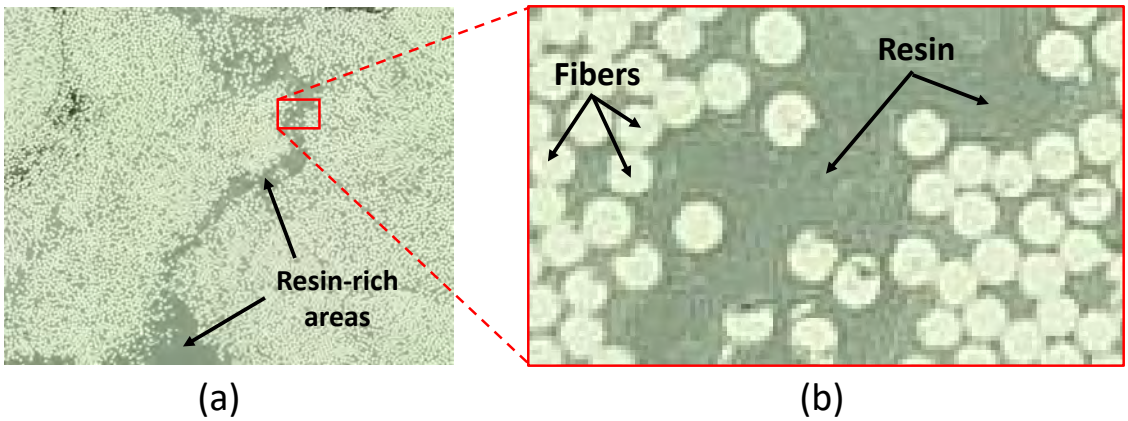

Figure 1.3: Transverse optical microscope image, and a close-up, showing a cross-section of fibers and resin.

The objective of this research is to automate the process of characterizing important microstructural properties of fiber-reinforced polymer (FRP) composites from their cross-sectional images by developing efficient and scalable geometric processing techniques. In this research, we will advance the state of the art in spatial partitioning algorithms; utilize those new algorithms to recognize fiber cross-sections and evaluate their breakage; and identify the resin-rich area defects and provide an explicit definition of their boundaries. The accurate identification of fiber cross-sections and resin-rich areas facilitate better understanding of composite microstructure.

1.2 Resin-rich areas' relation to composite properties

In FRP cross-sectional images, fiber cross-sections are usually identifiable from their shapes and pixel intensities, which enable us to analyze the geometric distribution of the fibers.

Researchers have identified the importance of analyzing the fiber geometry distribution, and its correlation to composite material properties and failure initiations [12, 13, 14]. One of the most common phenomena in the cross-sectional images is resin-rich areas, which are both easy to observe, and useful in the evaluation of material properties.

Yamashita et al. [15] demonstrate the relationship between resin-rich areas and volume resistivity of carbon-fiber tape reinforced thermoplastics. Smaller, less-frequent resin-rich areas leads to a higher possibility of contacting fibers, which improves the formation of electrically conductive paths, resulting in lower volume resistivity. Sacchetti et al. [16] found a positive correlation between the toughness of unidirectional Carbon/PEEK joints and the thickness of their resin-rich bond line areas. With a numerical approach, Ghayoor et al. [10] demonstrated that composite materials with resin-rich areas have lower stiffness and failure strain, compared to materials without resin-rich areas. According to a recent study by Ahmadian et al. regarding the failure response of carbon fiber reinforced polymers [17], although the presence of resin-rich areas does not significantly impact material strength and toughness under tensile and shear loads, it greatly affects such material properties under compressive loads.

Resin-rich areas may also indicate the edge/boundary of particular structures in the composites. For example, in ultra-thin chopped carbon fiber tape reinforced thermoplastics, resin-rich areas occur at the edge of the tapes [15]. In FRP 3D printed parts, resin-rich areas occur at the boundary between 3D printing layers. Therefore we can use resin-rich areas as special patterns to help identify these corresponding structures (tapes or 3D printing layers) in composite parts.

Since resin-rich areas have a close relationship to material properties and the aforementioned structures, it is useful to have an efficient algorithm to calculate their sizes and locations.

1.3 Thesis structure

This thesis is organized as follows. Chapter 2 reviews related literature regarding microstructural analysis from composite material cross-sectional images, and geometric preliminaries that will be used in this research.

Then, two approaches, the Voronoi approach and the distance transform approach, are described to detect resin-rich areas by using exact computation and sample-based techniques respectively.

The exact computation (Voronoi) approach is described in Chapter 3 and 4. In Chapter 3, a novel algorithm is proposed to automatically detect and categorize fiber cross-sections from cross-sectional images. With the exact locations, sizes and shapes of these detected fiber cross-sections as inputs, a novel algorithm is described in Chapter 4 to automatically detect resin-rich areas via constructing the Voronoi diagrams and α-shapes of the fibers. The newly proposed Voronoi and α-shape construction algorithm is specialized to exploit characteristics

typical of FRP 3D printed parts for significant efficiency gains. In this algorithm, the exact computation of Voronoi diagrams takes about 80% of the total computation time.

In Chapter 5, in order to accelerate the Voronoi construction process, a novel sample-based GPU algorithm is presented to construct Voronoi diagrams of spheres/circles. This chapter demonstrates that a new sample-based Voronoi diagram construction algorithm is able to achieve sufficient scalability and robustness compared to exact computation methods, but with a much shorter computation time. This chapter focuses on the design of a sample-based algorithm for the Voronoi diagram construction, which is only one step in the resin-rich area detection process. It is possible to design a sample-based algorithm for the whole process to further increase the computation efficiency.

This fully sample-based (distance transform) approach is described in Chapter 6. In this chapter, we apply the concept of α-hulls to formalize a mathematical definition of the boundaries of resin-rich areas, and design an efficient sample-based process to directly compute the defined boundaries in the discrete image space via the distance transform and morphological dilation operation. Compared to the exact computation (Voronoi) approach, this sample-based approach reduces the computation time by 95.3%

In chapter 7, we compare the advantages and disadvantages between our two defect detection approaches and discuss potential future research directions.

1.4 Statement of multiple authorship and prior publication

In this dissertation, the fiber recognition and categorization method presented in Chapter 3 is based on the paper [18] published in *IEEE International Conference on Image Processing (ICIP) 2021* conference; the GPU-based Voronoi construction algorithm presented in Chapter 5 is based on the paper [19] published in the journal *Computers & Graphics*; and the distance-transform-based resin-rich area detection presented in Chapter 6 is based on the paper [20] published in the journal *Composites Part B: Engineering*.

Although I am the primary author and implementer of the research mentioned above, this work could not have been done without the support from and collaboration with my advisor Prof. Sara McMains and my co-authors: Prof. Adarsh Krishnamurthy, Iddo Hanniel, and Sara Shonkwiler.

Chapter 2

Background and Related Work

2.1 Prior approaches to the detection of resin-rich areas

To calculate resin-rich areas from cross-sectional images, researchers have previously proposed algorithms based on the geometric concept of Voronoi diagrams. The Voronoi diagram is a spatial tessellation method to partition space into Voronoi cells corresponding to each input object such that any point inside the Voronoi cell is closer to its corresponding object than any other object.

To estimate the locations of resin-rich areas, researchers [10, 21, 22] first recognize fiber cross-sections from the FRP cross-sectional image, and then build a Voronoi diagram by treating centers of recognized fiber cross-sections as input objects. Since there is exactly one fiber cross-section in each Voronoi cell, the local "fiber volume fraction" inside each cell is $Area_{fiber}/Area_{cell}$, where $Area_{cell}$ is the size of the Voronoi cell and $Area_{fiber}$ is the size of its corresponding fiber cross-section. Thus, if the fiber cross-section areas ($Area_{fiber}$) are assumed to be about the same size (which these Voronoi-based algorithms typically do assume), then large-size Voronoi cells indicate local resin-rich areas (low fiber volume fraction areas). Large connected resin-rich areas can then be detected by merging neighboring large-size Voronoi cells.

Yang et al. [22] uses Voronoi diagrams to characterize the non-uniform distribution of fiber cross-sections, estimate the Voronoi cell sizes, and build a histogram of the estimated Voronoi cell sizes. Wide distribution of Voronoi cell sizes in the histograms could be used to indicate the existence of resin-rich areas. Ghayoor et al. [10] partition the cross-sectional image by the geometric dual structure of the Voronoi diagram (the Delaunay triangulation), identify large sub-regions by thresholding, and merge neighboring large-size sub-regions to form the resin-rich areas. Gommer et al. [21] tessellate the fiber bundle area into Voronoi cells, and calculate the local fiber volume fractions inside each cell. This information is then used to visualize the local fiber densities among the fiber bundle area and help people better distinguish the resin-rich areas. Although such Voronoi-based approaches are able

to provide a rough estimate of resin-rich areas, their applicability is limited because they assume equally-sized circular fiber cross-sections, which is usually not the case in practice [17, 23, 24], and they are orders of magnitude slower than our proposed approach based on α-hulls. Please refer to Section. 6.3.5 for more detailed discussion and comparisons.

2.2 The α-hull and its application

The α-hull, and its close relative, the α-shape, are powerful computational geometry concepts used to delineate boundaries of densely distributed input points. Both α-hulls and α-shapes are widely utilized in various science and engineering applications such as object reconstruction from 3D scanned data [25], which reconstructs an object from finite sample points on its surface or/and interior space; and molecular shape analysis [26], which determines the surface of protein molecules by treating the centers of their atoms as input points. The computation of α-shapes from point set inputs is available in software packages such as MATLAB [27] and CGAL [28].

However, for our application, we need to build α-hulls of fiber cross-sections that appear as arbitrary shapes instead of points. Since the α-hull concept was primarily designed for point set inputs, very few studies focus on the construction of the α-hull of arbitrary shapes. Several researchers have proposed algorithms to construct α-hulls of simple primitive shapes like circles and spheres [29, 30], but those algorithms cannot handle arbitrary shapes. It would be possible to implement a brute-force version of our approach that approximates α-hulls/α-shapes by building on functions available in existing software packages such as the alphaShape() function in MATLAB by treating each fiber cross-section pixel as an input point; however, this results in merely a pixel-wise approximation and moreover the overall computation time is impracticably long (e.g. over 85 minutes for each parameter choice for a real-world FRP cross-sectional image of 18,270*10,306 pixels; see A).

Therefore, a new algorithm is proposed to construct α-hulls from inputs of arbitrary shapes to identify resin-rich areas.

2.3 Preliminaries

In this section, we review the background of some geometric concepts that will be used in this research.

2.3.1 Voronoi Diagram, Dual Triangulation, α-Shape, and α-Hull

Given a set of objects $O = \{O_1, \ldots, O_n\}$ in $\mathbb{R}^2$, the Voronoi diagram of O is defined as the partition of the plane into n Voronoi cells, where each Voronoi cell is the set of all points closer to a particular input object O_i than to O_j ($\forall j \neq i$).

Within the Voronoi diagram, Voronoi edges are common boundaries between two adjacent Voronoi cells. Any point on a Voronoi edge is equidistant to its two corresponding input

objects. The intersections of Voronoi edges are Voronoi vertices. Each Voronoi vertex is equidistant to all of its corresponding input objects.

Because of this equidistance property, a Voronoi vertex always corresponds to the center of an empty circle that is tangent to all of the vertex's corresponding objects. Following the nomenclature for Voronoi diagrams of points, we call this a *circumcircle*, centered at the Voronoi vertex, having as its radius the distance between the Voronoi vertex and the corresponding objects. It does not overlap with the interior of any input object (see Fig. 2.1(b)).

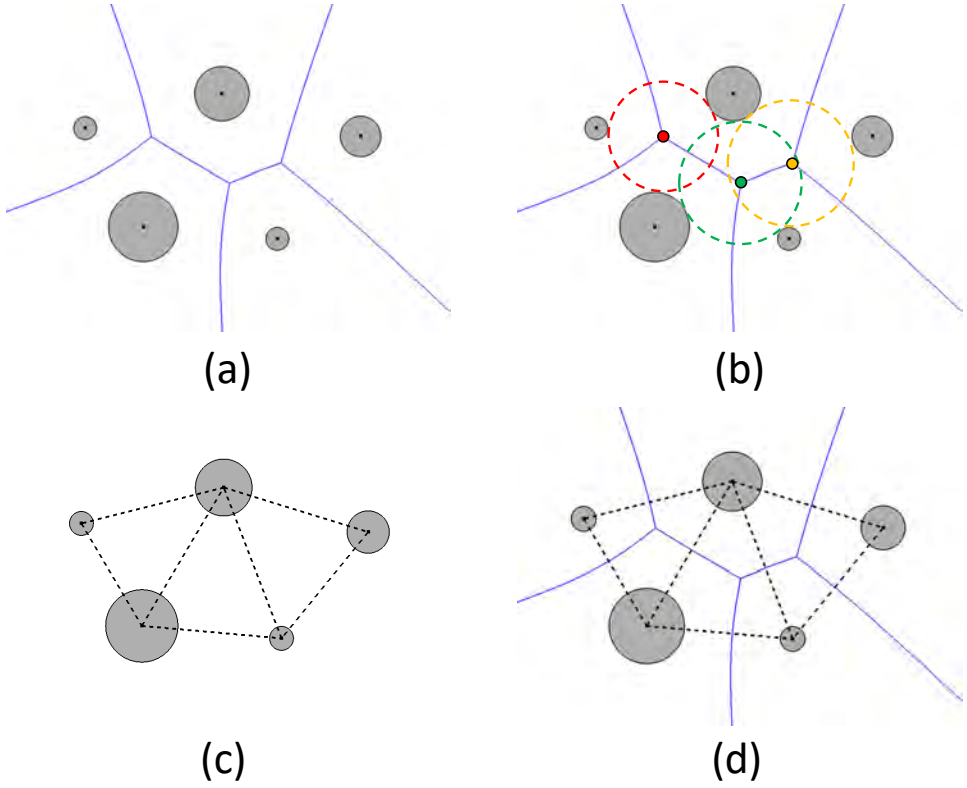

Figure 2.1: Voronoi diagram and dual triangulation of a set of input circles with different radii. (a) Voronoi diagram, (b) Voronoi vertices and corresponding circumcircles, (c) dual triangulation of the same input, (d) superimposed Voronoi diagram and dual triangulation, illustrating their duality.

The calculation of Voronoi vertex geometry is identical to the calculation of the circumcircle of a set of three corresponding objects. In our application, we focus on circle and ellipse inputs. We use the circumcircle calculations for circles and ellipses detailed in [31, 32] respectively.

The dual triangulation (see Fig. 2.1(c)) is the dual structure of the corresponding Voronoi diagram. It satisfies the following duality properties:

- Each Voronoi vertex corresponds to a bounded face in the dual triangulation. The corresponding circles or ellipses of the Voronoi vertex are the dual triangulation face vertices.
- Each Voronoi edge corresponds to an edge (perpendicular to it for circle inputs) in the dual triangulation.
- Each Voronoi cell corresponds to a vertex in the dual triangulation. Vertices in the dual triangulation are centers of the input circles or ellipses.

The relationship between a Voronoi diagram and the corresponding dual triangulation is illustrated in Fig. 2.1(d).

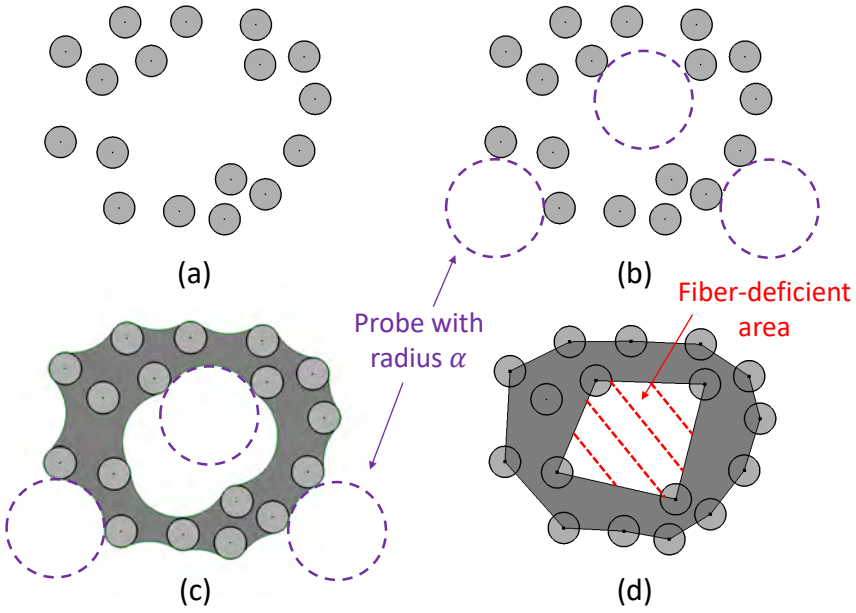

Figure 2.2: The α-hull and α-shape of circle inputs. (a) input; (b) probe with radius α moving around the space without colliding with any input; (c) α-hull; (d) α-shape.

The α-hull and α-shape are computational geometry concepts invented by Edelsbrunner et al. in 1983 [33] that was originally designed for point set inputs. In [34], Edelsbrunner and Mucke provide an intuitive description of the three-dimensional α-hull and α-shape:

> "Think of $\mathbb{R}^3$ filled with Styrofoam and the points of S made of more solid material, such as rock. Now imagine a spherical eraser with radius α. It is omnipresent in the sense that it carves out Styrofoam at all positions where it does not enclose any of the sprinkled rocks, that is, point of S. The resulting object will be called the α-hull. To make things more feasible we straighten

the surface of the object by substituting straight edges for the circular ones and triangles for the spherical caps. The obtained object is the α-shape of S."

Now apply this intuition to two dimensional circle and ellipse inputs: for a set of circle and ellipse inputs representing the fiber cross section (Fig. 2.2(a)), we use a probe with radius α to move around the whole 2D plane without overlapping any of the inputs (Fig. 2.2(b)). The union of regions where the probe cannot reach is the α-hull (Fig. 2.2(c)). We straighten the boundary of the alpha hull with straight edges, and obtain the α-shape (Fig. 2.2(d)). In our application, the α-hull or α-shape regions indicate the places where fibers are close to each other, since the probe cannot be placed without colliding with fibers, and the complementary regions of the α-hull or α-shape are the actual fiber deficient areas.

The α-shape can be constructed from the dual triangulation with the help of another concept: the α-complex. The α-complex is a simplicial complex that closely relates to the α-shape and the dual triangulation: it is a subset (subcomplex) of the dual triangulation where the elements (triangular faces and edges, also called simplices) are inaccessible by the α-probe; and the union of such elements is the α-shape (Fig. 2.3).

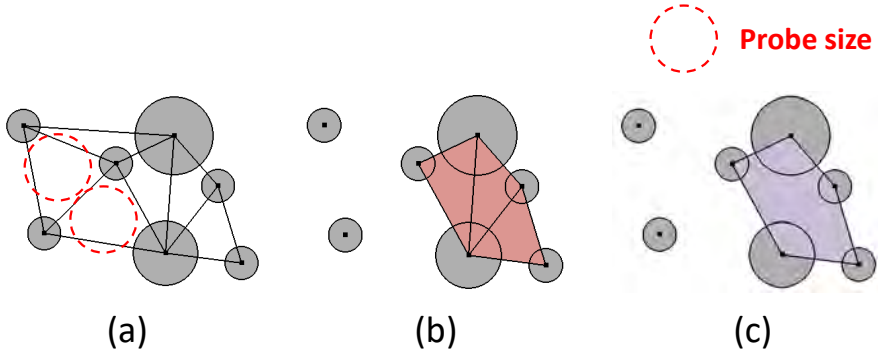

Figure 2.3: An example of the relationship among: (a) dual triangulation; (b) α-complex; (c) α-shape.

Note that the concepts of Voronoi diagrams and α-shapes were originally defined for point set inputs. Voronoi diagrams were then generalized to handle arbitrary objects, and these are sometimes referred to as "generalized Voronoi diagrams," but often "generalized" is omitted. Similarly, although we are generalizing α-shapes to handle arbitrary objects, we will just refer to them as α-shapes instead of "generalized α-shapes."

2.3.2 Distance transform

The distance transform is an image processing operation applied to binary images [35]. As illustrated in Fig. 2.4(a) and 2.4(c), a binary image includes foreground regions/pixels (1s) and background regions/pixels (0s). The distance transform takes the binary image as the

input, and returns a value for each pixel that is its distance to the nearest background pixel (Fig. 2.4(b)). The background pixels will have values of zeroes in the distance transform because they have zero distance to themselves; the foreground pixels will have values greater than zero and reflect how far they are from the nearest background pixel. The distance transform is usually visualized as a grayscale image where the pixel intensities show the corresponding distance values of each pixel (Fig. 2.4(d)). Pixels farther from the background regions have higher distance values and therefore higher intensities (brighter) in the grayscale image visualization of the distance transform.

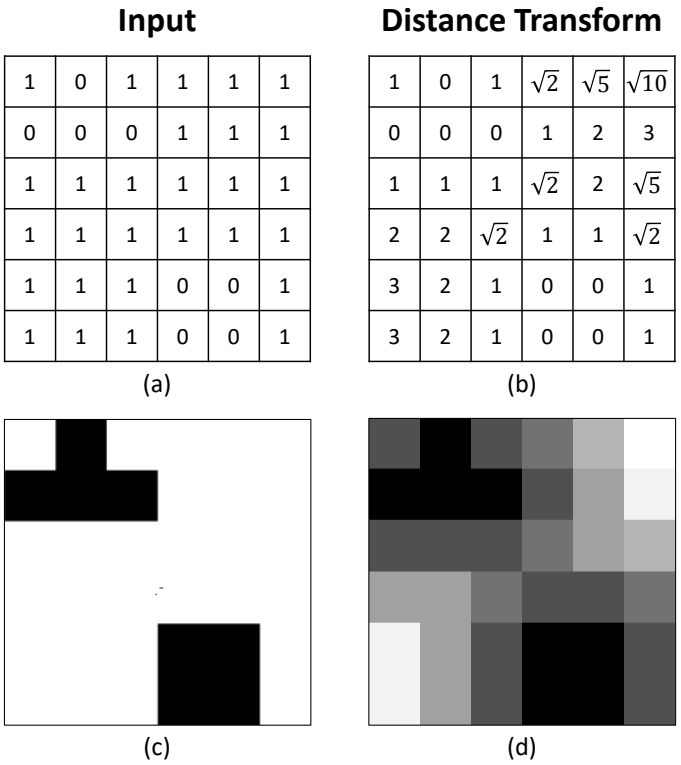

Figure 2.4: An example of the distance transform. The first row shows the matrix format: (a) the binary input and, (b) its distance transform; the second row visualizes these matrices: (c) the (same) input and, (d) its distance transform.

2.3.3 Morphological dilation

Morphological dilation (usually denoted as $\oplus$) is a basic operation in mathematical morphology. As illustrated in Fig. 2.5, for any two arbitrary 2D shapes A and B, the morphological dilation of A by B is defined as:

$$A \oplus B = \bigcup_{b \in B} A_b \tag{2.1}$$

where A_b is the translation of A by b. The translation of A by $b = (b_x, b_y)$ is further defined as:

$$A_b = \{c \mid c = a + b, \text{ for all } \ a \in A\} \tag{2.2}$$

where $c = (c_x, c_y) = (a_x + b_x, a_y + b_y) = a + b$.

The morphological dilation operation can be interpreted in a more intuitive way. From Fig. 2.5, if A is a random shape and B is a circle with center B_c, the dilation of A by B $(A \oplus B)$ is the union of all the areas covered by B when B_c moves inside A.

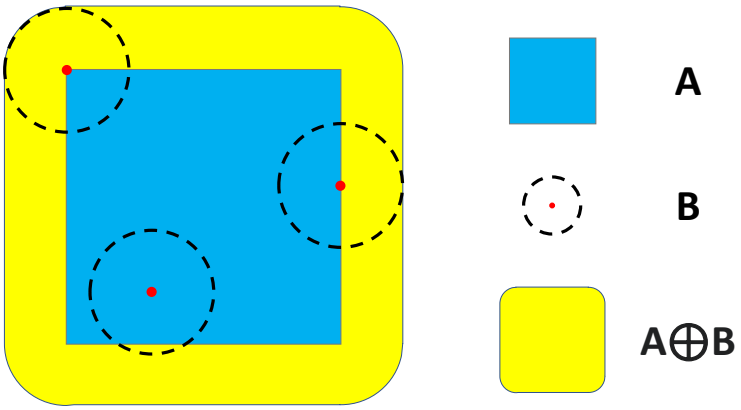

Figure 2.5: An example of the morphological dilation operation. Shape A is the blue square; structuring element B is the dashed black circle; and the morphological dilation of A by B $(A \oplus B)$ is the union of the yellow and blue areas.

In mathematical morphology of 2D images, A is usually an input binary image, and B is usually defined as a circle/probe that is called the structuring element.

2.4 Summary

In this chapter we have reviewed prior approaches to resin-rich area detection, the concept of α-hulls/shapes, and mathematical definitions of other (supporting) geometric concepts relevant to this research. In later chapters (Chapter 4, 5, and 6), we will show how α-hulls/shapes can be applied to define resin-rich areas, and how to construct α-hulls/shapes using those supporting geometric concepts. First, however, as inputs in the construction of α-hulls/shapes, the fiber cross-sections must be recognized from the microscope images.

In the next chapter, we will introduce a novel algorithm to automatically recognize fiber cross-sections, and evaluate their breakage.

Chapter 3

Fiber Recognition in Composite Materials

3.1 Introduction

As discussed in Section 1.1, because of the differing characteristics of the fiber reinforcements and polymer matrix, the geometric distribution of the fibers is the key factor for microstructural characterization of FRP composites. From their transverse microscope images, we are able to characterize the microstructure of FRP composites by recognizing fiber cross-sections and evaluating their breakage (Fig. 3.1). The effectiveness of the microstructural characterization relies on the accuracy of the fiber recognition process.

Furthermore, accurately recognizing fiber cross-sections is the prerequisite for resin-rich area detection. It is impossible to correctly detect areas where fibers are locally deficient without knowing the exact locations and sizes of these fibers.

Recognizing fibers from the microscope image usually follows a two-step process: (1) segment fibers that contact each other into individual fiber pixel blob regions; (2) fit a circle or an ellipse to each individual fiber pixel blob. Mlekusch [36] separated the touching fiber regions into individual fiber segments based on their convexity, and then fitted contour points

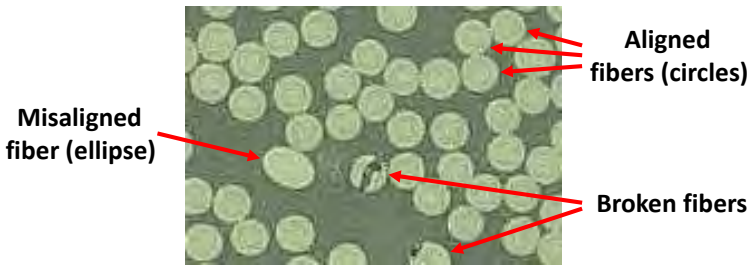

Figure 3.1: An example of a closeup of an input microscope image

of each segment into ellipses based on a regression calculation. In [37], Martin-Herrero et al. detected individual fiber blobs by using successive mathematical morphology operations, and then fitted ellipses to them based on least-squares orthogonal distance fitting [38]. Amjad et al. [39] applied a marker-controlled watershed segmentation to identify fibers, and then approximated them as ellipses by a Hough-transform-based ellipse detector [40]. The above methods performed well under the assumption that all the fibers are undamaged fibers (circles or ellipses); however, broken fibers are common in FRP composites (Fig. 3.1). An accurate method to recognize both the broken and complete fibers is necessary and is crucial to the overall evaluation of the composite material.

In this chapter, we present a novel method to automatically recognize fibers in composite material cross-sectional images, whether they are aligned or misaligned, complete or broken. This method first segments the binarized input image into individual fiber pixel blobs using watershed segmentation, and then identifies their breakage and fits circles or ellipses to them. To better identify the fiber breakage, we introduce a new method, called contour gradient charts, that efficiently distinguishes complete and broken fibers and further helps with the circle/ellipse fitting.

3.2 Fiber blob detection

Since the fiber cross-sections all appear as complete or broken circular/elliptical blobs in the input image, we first detect blob areas for each individual fiber. Fig. 3.2 illustrates the steps of our fiber blob detection process: (a) converting the input image to grayscale; (b) binarizing the grayscale image and removing noise; (c) building the distance transform from the binary image; (d) applying watershed segmentation on the distance transform; (e) merging oversegmented regions.

Beginning with the original microscope image, we convert it to grayscale and apply Gaussian smoothing to reduce noise (Fig. 3.2(a)). In the smoothed grayscale image, fiber pixels have observably higher intensity values than the resin matrix pixels. Based on the intensity contrast between fiber and matrix pixels, we convert the grayscale image into a binary image using Otsu's histogram-based global thresholding method [41] and denoise it (Fig. 3.2(b)). We do so by identifying connected fiber (white in the image) pixel regions below a minimum number of pixels. In our experiments, we found removing regions with fewer than 10 connected pixels (about 10% of fiber cross-section size) led to good fiber detection results.

After denoising, each individual connected fiber pixel region in the binary image corresponds to one of the following: an individual fiber, a group of contacting fibers, or a chunk of broken fiber. To segment individual fiber blobs from regions of contacting fibers, we exploit distance-transform-based watershed segmentation, as follows.

From the binary image, we build a Euclidean distance transform [42] that assigns each pixel its distance to the nearest polymer (matrix) pixel (Fig. 3.2(c)). The local maximum values in the distance transform efficiently indicate the centers of each individual fiber. We

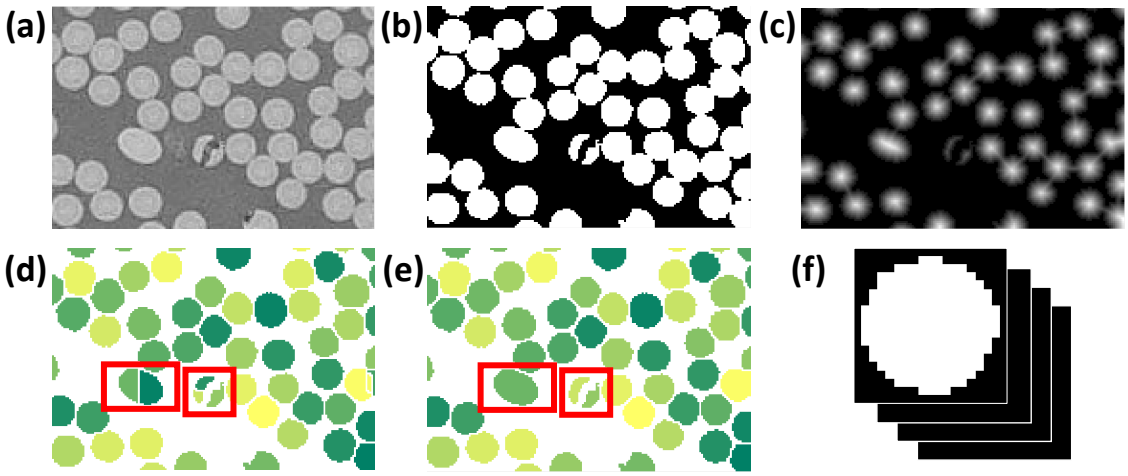

Figure 3.2: The procedure for fiber blob detection: (a) grayscale; (b) binarization and cleaning; (c) distance transform; (d) watershed segmentation; (e) merging oversegmented regions; (f) result: individual fiber pixel blobs.

then run watershed segmentation [43] on the distance transform, separating contacting fibers into individual fiber blobs (Fig. 3.2(d)).

In the continuous space, perfect circles and ellipses only have one local maximum distance point at their centers. However, since the composite images are represented in discrete pixels, the fibers are not perfect circles or ellipses, which may cause multiple local maxima being calculated near the fiber center point. As illustrated in Fig. 3.2(d) left red box, this can cause an oversegmentation in the watershed segmentation. Broken fibers may also be oversegmentated because of their irregular shapes (Fig. 3.2(d) right red box). To deal with these oversegmentation problems, after the watershed segmentation, we merge detected regions whose local maxima in the distance transform were close and that were also connected in the binary image (Fig. 3.2(e)).

The output of the fiber blob detection is pixel blobs representing detected individual fibers (Fig. 3.3). Next (Section 3.3), we will describe how we identify their breakage, and compute their locations accordingly.

3.3 Fiber breakage evaluation and localization

After identifying the pixel blobs for each fiber, we propose a novel tool called contour gradient charts to help determine if they are broken. Based on the analysis from the contour gradient charts, we select unbroken contour pixels and fit a circle or ellipse to each fiber.

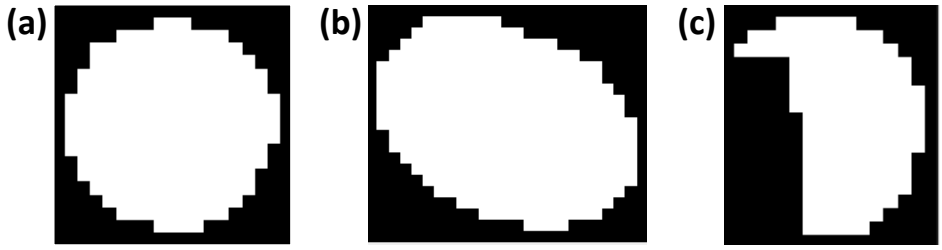

Figure 3.3: Real-world examples of different fiber blob types: (a) complete aligned fiber; (b) complete misaligned fiber; (c) broken fiber.

3.3.1 Breakage evaluation via Contour Gradient Charts

Our approach to determining whether a fiber blob is broken or not is based on a simple yet powerful idea: for complete fiber blobs, the gradient direction along the boundary varies smoothly along all its contour pixels; for broken fiber blobs, the gradient direction varies smoothly only along the contour pixels bounding its unbroken portion, but rapidly changes at the broken portion of the contour pixels. This property enables us to determine if a fiber blob is broken by checking if rapid gradient changes exist along its contour pixels.

We define the gradient direction of a boundary point (assuming a continuous perfect circle for illustration) to be perpendicular to its tangent line and into the circle. An example is illustrated in Fig. 3.4, where twelve points are evenly sampled on the circle in clockwise order (from P_1 to P_{12}), with gradient directions for examples P_1, P_5, and P_{10} shown.

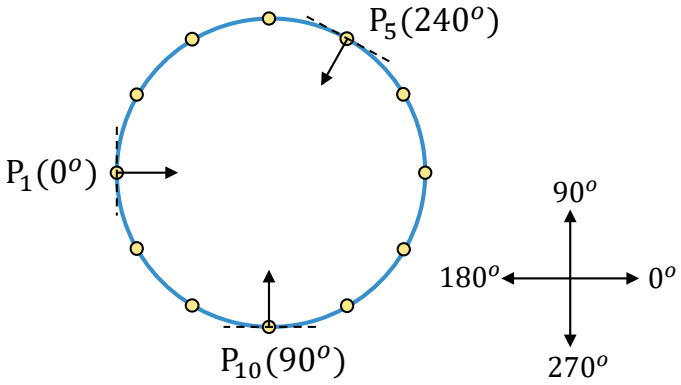

Figure 3.4: An example of contour sample points and their gradient directions.

We create a "Contour Gradient Chart" (CGC) to illustrate the gradient direction change along the boundary sample points. In the chart, x values are the sample points on the contour of the circle in clockwise order and y values are their corresponding gradient direction

angles (Fig. 3.5(a)). To analyze the rate of change of gradient angles, we take the derivative (numerical gradient) of the CGC modulo 360°and call it the "first derivative CGC" (Fig. 3.5(b)).

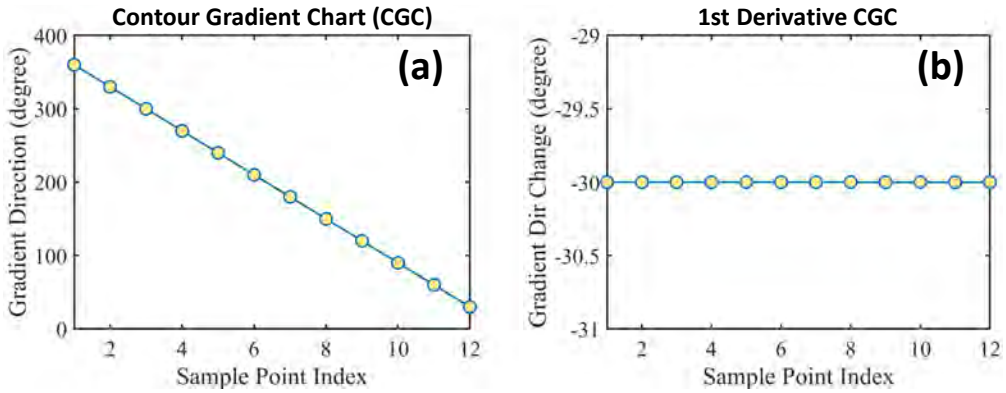

Figure 3.5: Contour gradient charts of the example in Fig. 3.4.

Now considering the fiber blobs detected from Section 3.2, we construct CGCs of each by selecting their contour (boundary) pixels as the sample points. For these contour pixels, we use the Sobel-Feldman operator to calculate their gradient vectors. The discretized contour pixels bring noise to the calculation of the gradient, but after applying a moving average operation, the gradient direction changes between sample points are small and approximately constant because of the smoothness of complete circles and ellipses (Fig. 3.6). Call the number of sample points N. The sum of all the signed gradient direction changes will total 360° along all boundary pixels, so the average value of gradient direction changes, which should be close to the approximately constant value for unbroken fibers, is $-360°/N$ in the first derivative CGC.

However, for broken fibers, dramatic gradient direction changes occur at the break points. It is easy to observe and classify the broken fibers and complete fibers from their first derivative CGCs by detecting if there are large gradient direction changes or not (Fig. 3.6 and 3.7). Furthermore, the unbroken portion of the contour can be identified using the first derivative CGC. Similar to a complete fiber blob, the corresponding range of the unbroken portion of a broken fiber would be seen as a long period of approximately constant gradient changes around $-360°/N$ in the first derivative CGC (Fig. 3.7).

3.3.2 Circle and ellipse fitting

The final step is to fit a circle or an ellipse to the candidate contour pixels of each fiber blob. For complete fibers, all of their blob boundary pixels are taken as the candidate pixels; for broken fibers, only the selected unbroken portion of boundary pixels are the candidate pixels.

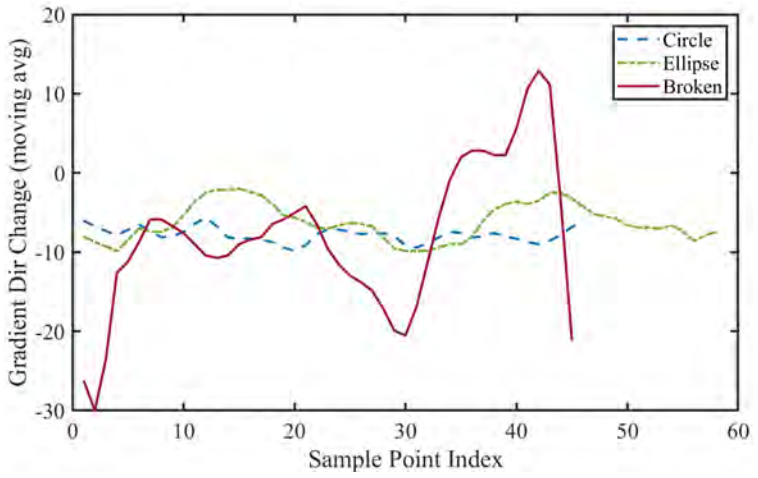

Figure 3.6: Example first derivative CGC of fiber blobs from Fig. 3.3. The circular, elliptical, and broken fibers have N_C=46, N_E=58, and N_B=45 boundary pixels, respectively. Complete fibers (circle/ellipse) have steady gradient direction changes close to $-360^\circ/N_C = -7.8^\circ$ or $-360^\circ/N_E = -6.2^\circ$; the broken fiber has dramatic changes exceeding $-360^\circ/N_B = -8^\circ$.

Since a circle is a special case of an ellipse, we choose to apply a direct ellipse fitting algorithm [44] to our candidate pixels in order to simultaneously handle both aligned and misaligned fibers (Fig. 3.8).

3.4 Experimental results

We tested our proposed method on more than 30 real-world microscope cross-sectional images from 3D printed FRP parts. The images have an average size of 18,000*9,500 pixels and contain about 500,000 carbon fiber cross-sections, with 7-8% broken fibers. The categorization and localization (the calculation of fiber boundaries) output of our algorithm on the test images was manually examined by material experts, who found no errors with our method's results for almost all fibers (>99.9% overall correctness, >99% correctness for broken fibers). Typical classification failures occur when the breakage is fully inside a fiber, or an inclusion with nearly smooth contour exists (Fig. 3.10(a, c)). In such cases, since no rapid gradient changes exist along the fiber boundaries, our algorithm falsely identifies them as complete fibers (Fig. 3.10(b, d)).

To the best of our knowledge, our approach is the first method that can robustly recognize broken fibers and calculate their fitting circles/ellipses. In contrast, direct circle/ellipse detection methods [40, 45] are generally based on the theory of the Hough transform, which is sensitive to noise. Because there is often fiber debris (noisy pixels) around the breakage locations, the direct detection methods are prone to generate false positive detections around

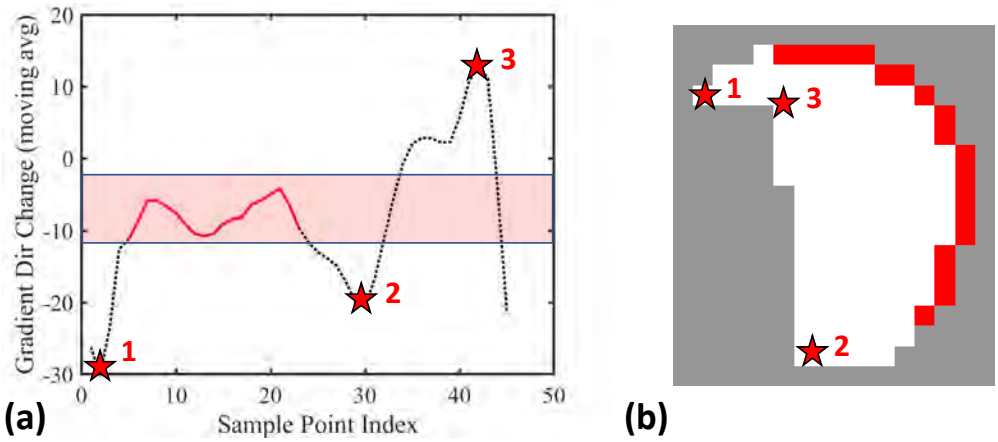

Figure 3.7: Selected unbroken portion of contours are shown in red in (a) first derivative CGC, and (b) its corresponding fiber blob. Red stars demonstrate dramatic gradient direction changes and their relations to the actual break points.

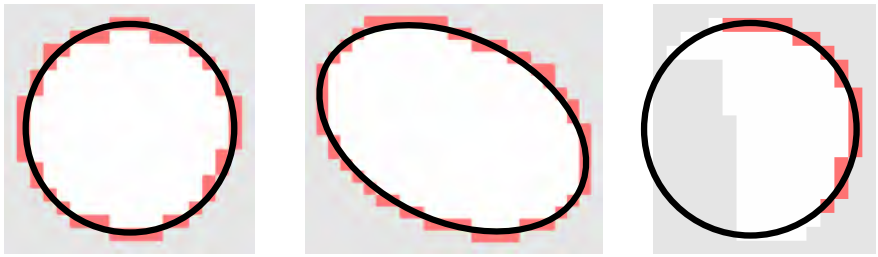

Figure 3.8: Circle/ellipse fitting results for different fiber blob types. The input pixels are rendered in light red.

the broken fibers (Fig.3.11(b)). Fiber blob segmentation and ellipse fitting methods [39, 36, 37] efficiently eliminate the noisy pixels by the blob segmentation process, but they are unable to recognize breakage in the fibers. The contour pixels on the broken portion may cause inaccurate circle/ellipse fitting (Fig.3.11(c)). Our proposed method only selects contour pixels on the unbroken portion after eliminating noisy pixels during the fiber blob detection process, and only then fits ellipses to identify misaligned fibers. These operations lead to a more robust and accurate detection of the broken fibers (Fig.3.11(d)).

3.5 Conclusions

In this chapter, we propose a novel fiber recognition algorithm, for cross-sectional microscope images of composite materials that detects all types of fibers and distinguish which are

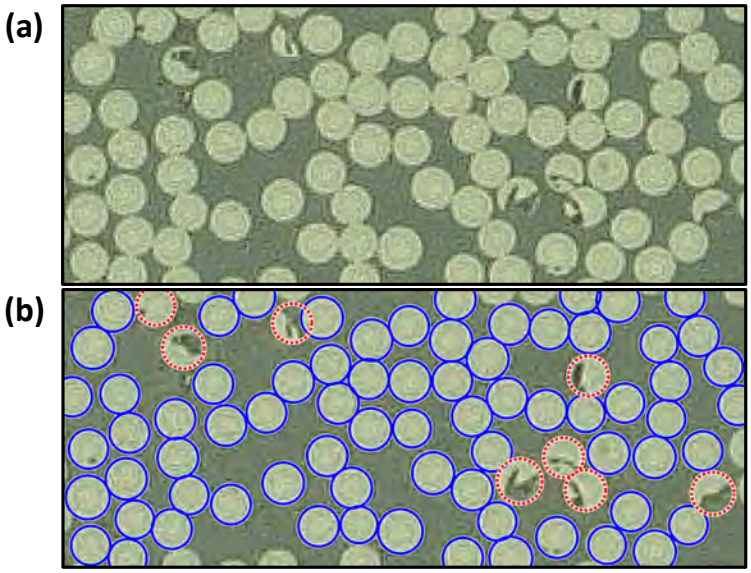

Figure 3.9: Experimental results: (a) the input image; (b) recognized complete (blue) and broken (red dotted) fibers.

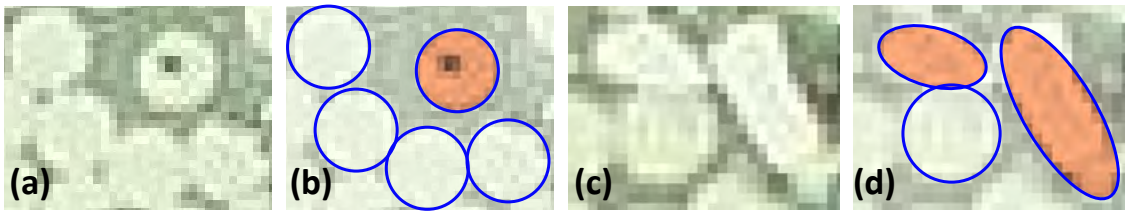

Figure 3.10: Failure cases: (a-b) a broken fiber with only inner breakage is falsely identified as a complete fiber; (c-d) (two) inclusions falsely identified as complete misaligned fibers. (Incorrect identifications are shaded red-beige.)

broken, unbroken, and/or misaligned. We introduce contour gradient charts, which enable identifying broken fibers as well which boundary pixels are on their unbroken contours, for more accurate circle and ellipse fitting. The performance is validated on real-world microscope images from 3D-printed fiber-reinforced polymer parts, on which our method is able to correctly identify and classify over 99.9% of fibers.

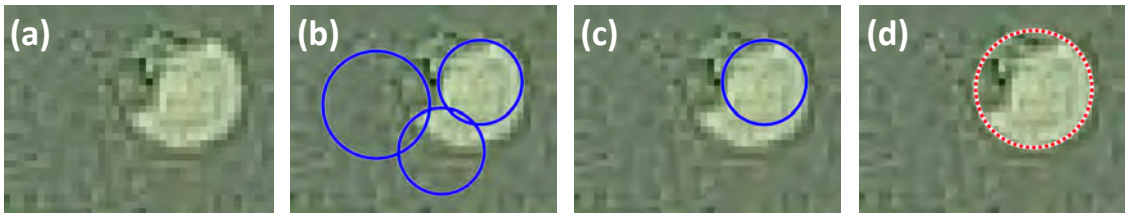

Figure 3.11: Comparison of results of our method and other popular methods on a broken fiber: (a) input; (b) direct circle/ellipse detection [40, 45]; (c) fiber blob segmentation and ellipse fitting [39, 36, 37]; (d) our method.

Chapter 4

Defect Detection from Microscope Images: Voronoi Approach

4.1 Introduction

In this chapter, we present a novel algorithm to automatically detect resin-rich areas from microscope images of 3D printed FRP parts. This algorithm takes the result of the fiber recognition process (as described in Chapter 3) as the locations of fibers, and determines the resin-rich areas from the input circles and ellipses using their Voronoi diagram and corresponding α-shape [33]. Our main contributions include:

- A novel approach to automatically find resin-rich areas in microscope images of 3D printed FRP parts using α-shapes. Both aligned and misaligned fibers are considered.
- Fast computation of the Voronoi diagram and alpha-shape of circles and ellipses using an approach specifically designed to exploit the characteristics of FRP composites. It is especially efficient when the majority of the fiber cross-sections are of similar size.
- Validation of our method on real-world microscope images. Our algorithm robustly detects resin-rich areas in real-world microscope images containing 100,000 fibers in 3.5 s, and 1,000,000 fibers in 70 s (excluding the time taken for circle and ellipse detection, which depends strongly on the resolution of the microscope image).

4.2 Related work

4.2.1 Voronoi diagram of circles and ellipses

Multiple algorithms have been proposed to calculate the Voronoi diagram of circles. Kim et al. proposed an algorithm that uses the ordinary Voronoi diagram of the circle centers as a seed, and then updates its topology by a series of edge-flipping operations [46, 31]. Jin

et al. report a sweepline algorithm that handles circle inputs in arbitrary locations. The input circles are allowed to intersect or even fully contain each other [47]. Lee et al. designed an efficient topology-oriented incremental algorithm that robustly constructs the Voronoi diagram of 100,000 input circles in seconds, while handling degenerate cases [48]. Some approaches for constructing the 3D Voronoi diagram of spheres, such as region expansion [49] or GPU ray-casting [50, 19], have 2D analogs for efficiently computing Voronoi diagrams of circles.

Beyond circles, the method of Emiris et al. [32, 51] can construct Voronoi diagrams of both circles and ellipses. However, it takes about 60 s to process 200 input ellipses [51], and is impractical for applications with larger numbers of inputs.

Currently, no existing methods can efficiently construct the exact Voronoi diagram of inputs with large numbers of both circles and ellipses. Some approaches are efficient for circle inputs [46, 31, 47, 48], but are difficult to extend to ellipse inputs; others handle both circle and ellipse inputs [32, 51], but are slow for large numbers of inputs.

Our inputs are predominantly composed of circles of essentially the same size with very few exceptional ellipses and large-size circles, and we have designed an efficient method to calculate exact Voronoi diagrams of inputs having these properties.

4.2.2 The α-shape of objects and its application

The α-shape is a computational geometry concept originally introduced for point set inputs [34] that captures the shape that could be accessed by a probe of radius α without intersecting the inputs. Generalizing α-shapes from point inputs to spheres, Kim et al. elucidated the relationship between the α-shape of input consisting of spheres of different radii, their Voronoi diagram, and its dual triangulation, and proposed efficient algorithms for their construction [52]. Inspired by this approach, we have developed a new method to construct the α-shape of input comprising circles and ellipses, and use its complementary regions to determine the resin-rich areas.

4.3 Algorithm overview

The main idea of our algorithm is to use α-shapes to evaluate the proximity among the cross-section of fibers in the microscope image, and detect the areas where fibers are locally deficient.

Fig. 4.1 illustrates the steps of our algorithm:

0. (Preprocessing). Detecting the locations and sizes of complete fiber cross sections in the input microscope image, using the method described in Chapter 3).
1. Constructing the Voronoi diagram based on the fiber geometries.
2. Constructing the dual triangulation from the Voronoi diagram.

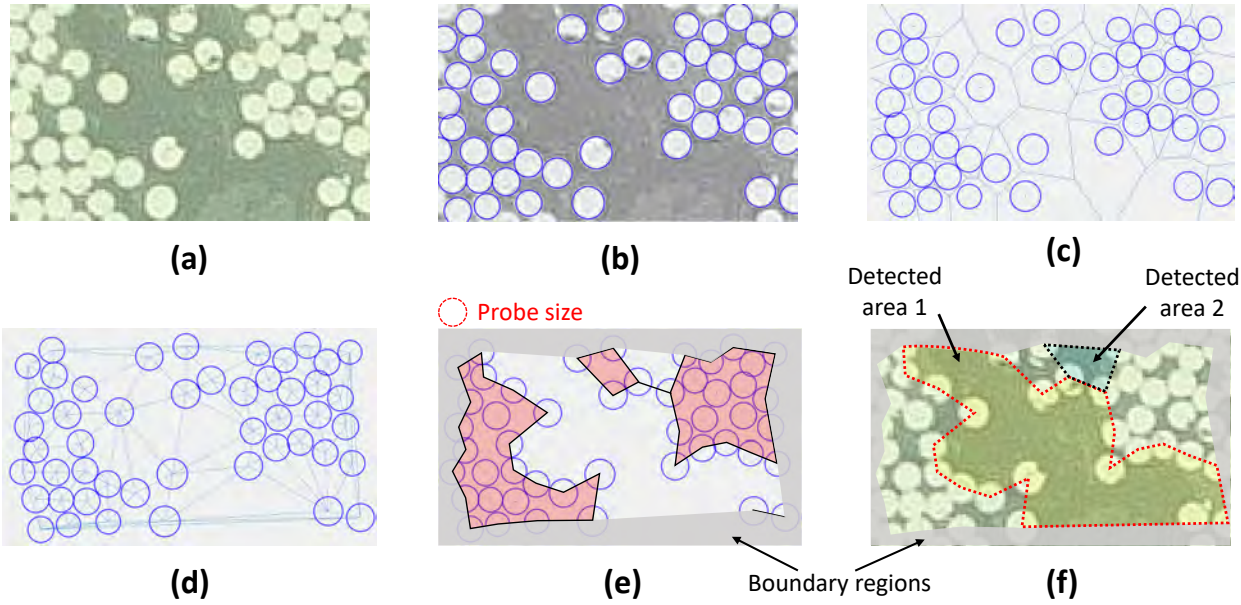

Figure 4.1: Algorithm overview: (a) input microscope image, (b) circle detection, (c) Voronoi diagram, (d) dual triangulation, (e) α-shape in the refined region, (f) resulting resin-rich areas, the complement of the α-shape. The boundary region is the complement of the refined region, and is ignored.

3. Calculating the refined region to focus on (as well as its complement boundary region). Build the α-shape from the dual triangulation in the refined region, identifying areas where fibers are close together.

4. Calculating the complement of the α-shape, giving the resin-rich areas. Further check and divide them into sub-areas if necessary.

Transverse cross-sectional microscope images of 3D printed FRP materials have the following characteristics:

- A typical image contains hundreds of thousands of fiber cross-sections.
- Most fibers are aligned, appearing as circles of about the same size, with a few appearing larger. Only a few fibers are misaligned, appearing as ellipses.

Our algorithm is designed specifically for these input characteristics. The construction of the Voronoi diagram and α-shape is designed for efficiency for inputs consisting primarily of identical radius circles, with a relatively small number of ellipses and circles of large radii.

4.4 Construction of the Voronoi diagram

4.4.1 Approach

Preprocessing the microscope image provides a set of circles and ellipses representing the locations and sizes of the detected fibers. In the next step, we build the Voronoi diagram of these circles and ellipses.

Recall that most fibers have circular cross-sections with identical radius, with a few exceptions having larger radius or are elliptical. For convenience, we will call the majority, identical circles *regular* input sites, and the exceptional size circles or ellipses *special* input sites. The input sites may slightly overlap each other, but none are fully contained within others (i.e. there are no hidden sites).

The construction of the Voronoi diagram follows a simple idea: we first build a Voronoi diagram assuming all inputs are regular input sites. We then expand the Voronoi cell of each special input site, one after another, gradually transforming the preliminary Voronoi diagram to the final Voronoi diagram. We call this process the cell-expansion method.

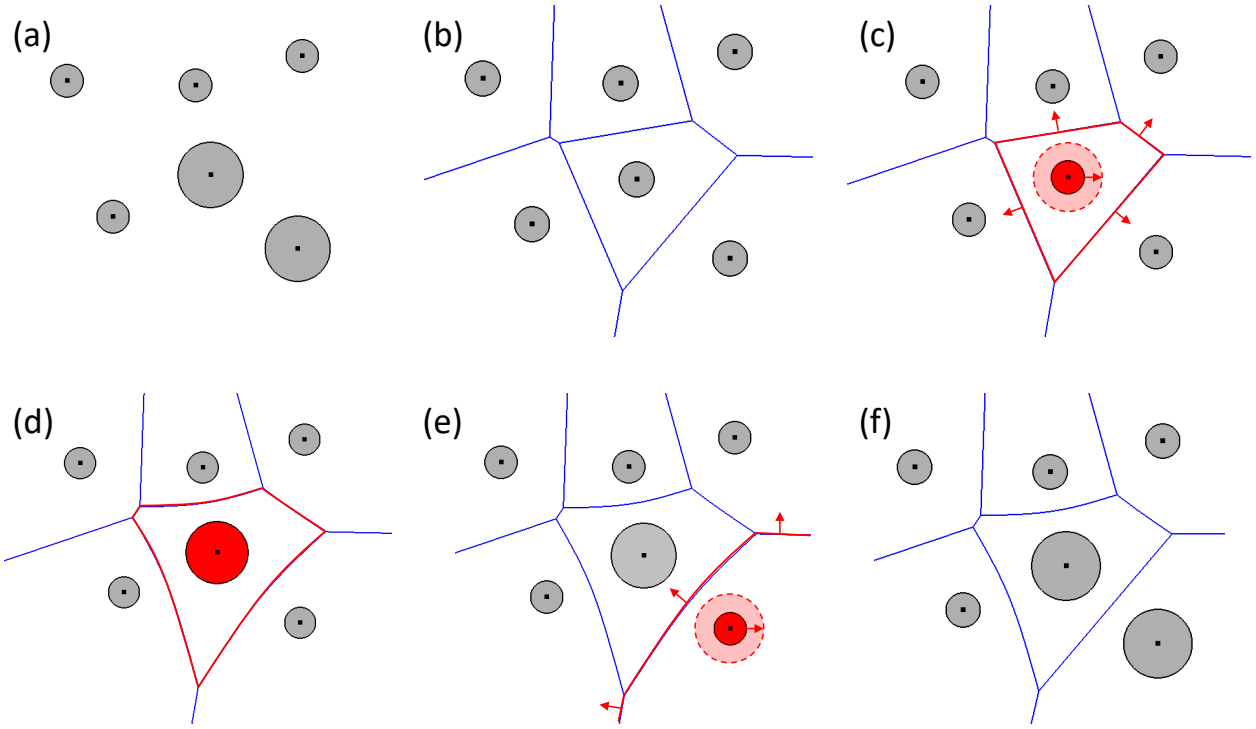

Figure 4.2: Cell-expansion process: (a) input, (b) ordinary Voronoi diagram, (c,d) expanding the Voronoi cell of one special site, (e) expanding the Voronoi cell of the second special site, (f) result. The expanding input sites and Voronoi cells are in red.

Fig 4.2 shows an example of the serial cell-expansion process. The input consists of four regular sites and two special sites (see Fig. 4.2(a)). We first calculate the Voronoi diagram assuming all input circles are regular sites (see Fig. 4.2(b)). Constructing the Voronoi diagram of a set of identical circles is the same as constructing the ordinary Voronoi diagram of a set of points; in our implementation, we use the Triangle library by Shewchuk [53] to do so. This library uses a data structure that simultaneously represents the ordinary Voronoi diagram and its dual triangulation. Starting from the ordinary Voronoi diagram, we iteratively expand the Voronoi cells of each special site in random order, updating the cells' geometry and topology in the data structure (see Fig. 4.2(c–e)), until all special sites have been processed (see Fig. 4.2(f)). Note that Voronoi edges that are straight line segments for equi-sized circle input generally become curved when their neighbors are special sites.

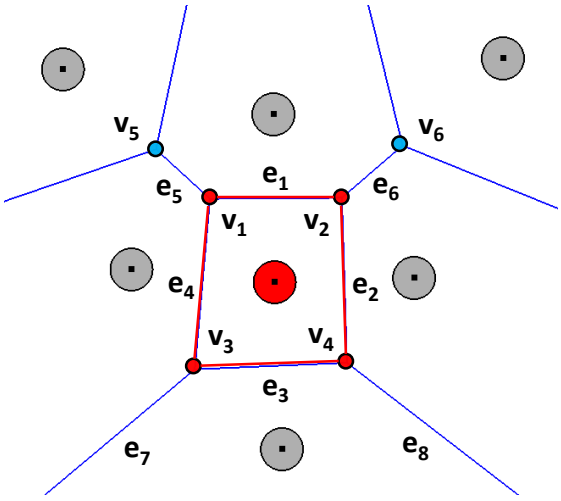

Figure 4.3: An expanding Voronoi cell (red) with: cell vertices v_1–v_4, cell edges e_1–e_4, radiating edges e_5–e_8, and neighboring vertices v_5, v_6.

In Fig. 4.3, suppose we are extending the red circle and its Voronoi cell. Its *cell vertices* are v_1–v_4, and its *cell edges* are e_1–e_4. Voronoi edges connected to cell vertices, but not cell edges, are *radiating edges* of the expanding cell: e_5–e_8. Voronoi vertices connected to radiating edges but not cell vertices are *neighboring vertices*of the expanding cell: v_5 and v_6).

Some Voronoi cells are unbounded, with Voronoi edges having one or both or ends at infinity. We call such edges *infinite cell edges*; we treat them as connected to distinct cell vertices at infinity. An example of an unbounded Voronoi cell is shown in Fig. 4.4. Suppose we are expanding the red circle and its Voronoi cell. Then e_3 and e_4 are infinite cell edges connected to different cell vertices at infinity, v_4 and v_5 respectively.

As we increase the size of a particular input site, its corresponding Voronoi cell expands accordingly. Kim et al. showed that for 3D Voronoi diagrams of spheres, during the Voronoi

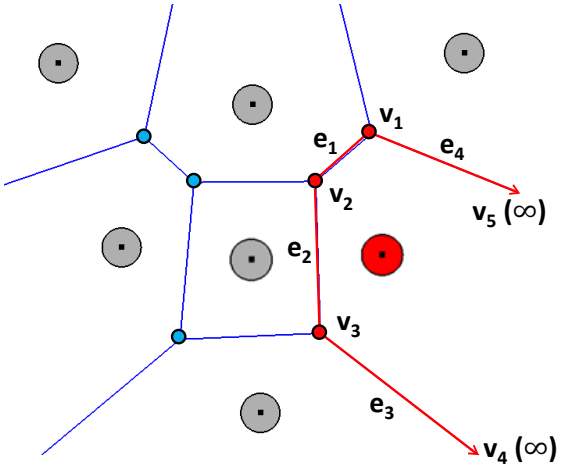

Figure 4.4: An expanding unbounded Voronoi cell (red) with: regular cell vertices v_1–v_3, cell vertices at infinity v_4, v_5; regular cell edges e_1, e_2, and infinite cell edges e_3, e_4.

region expansion process, topology changes only occur at Voronoi vertices or edges. Thus, it is sufficient to only consider the status of the edges and vertices of the expanding Voronoi cell for topology changes [49]. We state an analogous theorem for the 2D Voronoi diagram of circles and ellipses:

Theorem 1. *In a 2D Voronoi diagram of circles and ellipses, when a specific Voronoi cell is expanded, topology changes only occur at cell vertices, but not at cell edges.*

Proof. Suppose the theorem is false, i.e. topology changes may also occur at cell edges. During the expansion process, such cell edges would intersect other Voronoi cells, generating new Voronoi vertices on the edge. Consider Fig. 4.5. As cell 1 expands, its cell edge (red) intersects another Voronoi cell (cell 2), locally dividing cell 3 into cells 3 and 3', and generating a new Voronoi vertex. As explained in Section 2.3.1, the new vertex has a corresponding circumcircle that is tangent to the objects in each of its generating cells. So in this case, the input object of cell 3 must have tangent points to the circumcircle for both cell 3 and cell $3'$. However, this contradicts our assumption of convex inputs (circles and ellipses), since a convex shape only has at most one tangent point to a circle. Hence, the theorem is valid. □

During the serial cell-expansion process for all special sites, during the expansion of each special site, our algorithm follows a two-step process to detect potential vertex status and topology changes:

1. Check if the expanding site has infinite cell edges, which may generate new vertices.

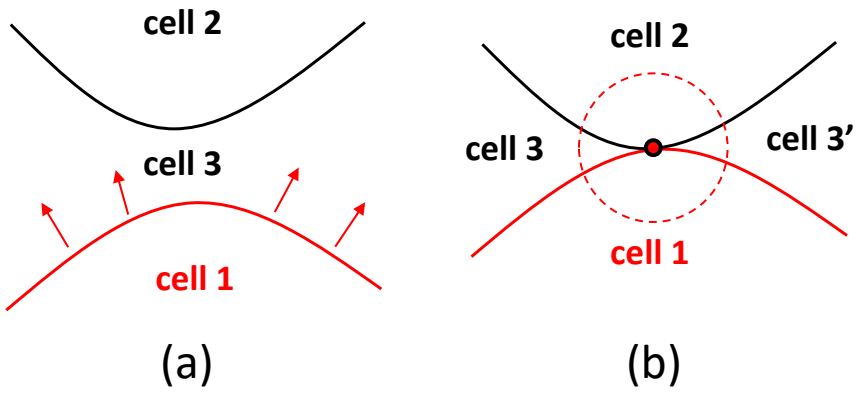

Figure 4.5: Topology changes at cell edges during expansion. This situation only exists for non-convex inputs, and not for circle and ellipse inputs.

2. Next, for each cell vertex, check its interaction with neighbors by tracing along radiating edges. Each vertex might collide with a neighbor vertex, disappear, or remain unchanged in its topology.

We now explain each step in detail.

4.4.2 Checking infinite cell edges

During the cell-expansion process, if the current expanding site has an unbounded Voronoi cell, we check all of its infinite cell edges to detect their potential intersections with infinite edges not belonging to the expanding cell.

In Fig. 4.6(a), consider infinite cell edge e_1. As the input site C_1 expands, e_1 moves towards its neighboring Voronoi cell corresponding to C_2 and potentially intersects with another infinite edge e_3. Edges e_2 and e_3 are shared by sites C_1-C_2, and C_2-C_3, respectively. If they intersect, there must be a new circumcircle generated by C_1, C_2, and C_3. Applying the method described in [31] (or [32] for ellipses), we find circumcircles from sites C_2, C_3, and the expanded C_1. As illustrated in Fig. 4.6(b), in this case, the newly generated circumcircle is detected from the calculation. From the existence of the new circumcircle, we confirm the intersection of e_1 and e_2, and then update the geometry and topology of the Voronoi diagram (Fig. 4.6(c)).

A disconnected edge is a special infinite edge both of whose ends extend to infinity. Following the process described above, we check for newly generated vertices at each of its ends. As shown in Fig. 4.7, new vertices may be generated at both of a disconnected edge's ends.

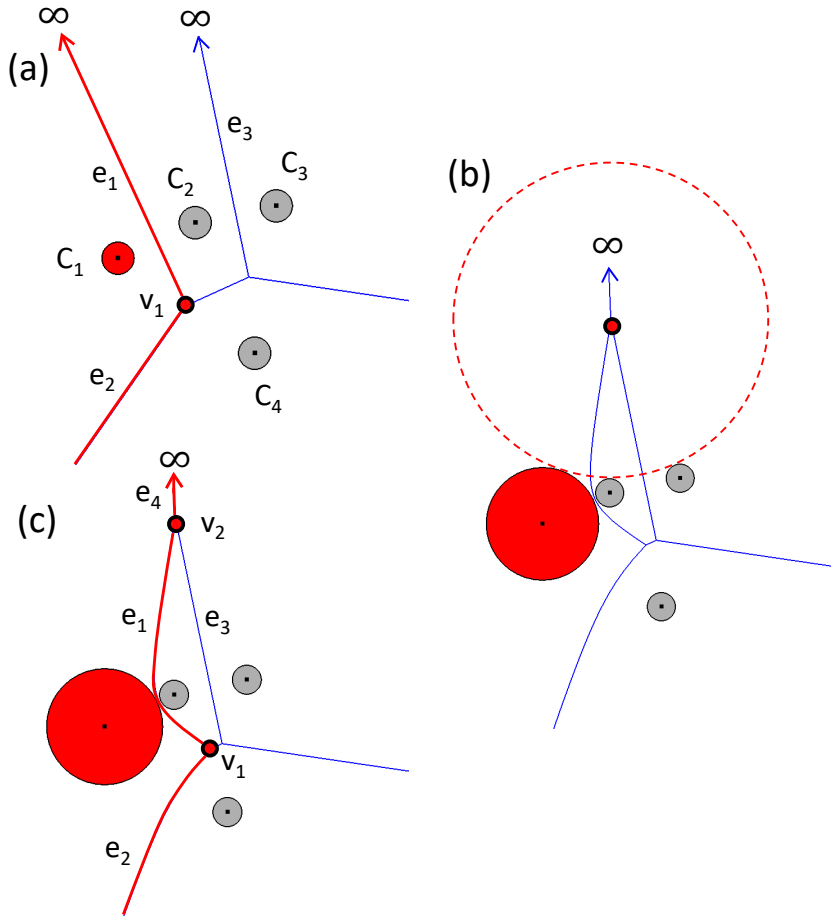

Figure 4.6: An expanding unbounded Voronoi cell with infinite cell edges. (a) Before expansion, (b) a newly generated circumcircle exists, (c) after expansion, with a new generated vertex v_2 and edge e_4.

4.4.3 Tracing cell vertices along radiating edges

For each special input, after checking its infinite cell edges and updating the topology and vertex geometry accordingly, we obtain an updated list of its cell vertices. When a Voronoi cell expands, each of its cell vertices moves outward along its corresponding radiating edge because the vertex must remain equidistant from the two sites generating the radiating edge. In this second step of the cell expansion process, we trace each of the non-infinite cell vertices along its radiating edge, detecting potential topology changes. We consider two cases.

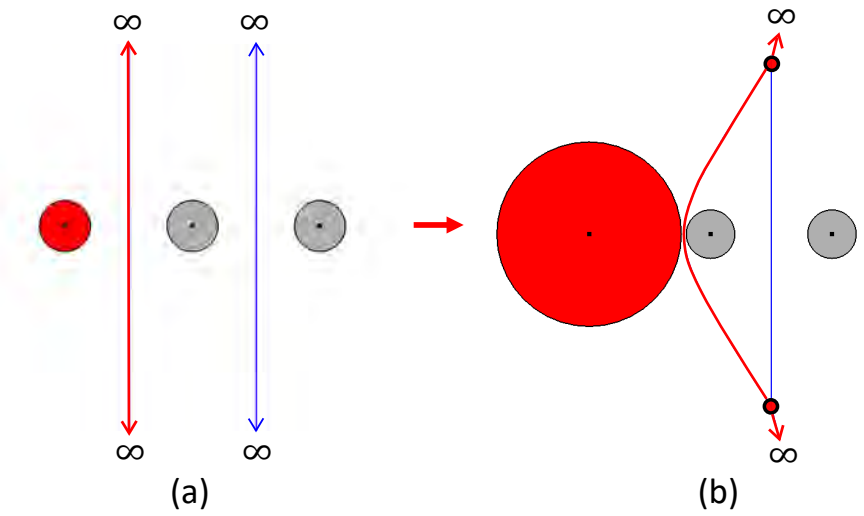

Figure 4.7: Disconnected edges in the cell-expansion process

Radiating edge extends to a neighbor vertex

For each cell vertex we are tracing, if its radiating edge extends to a neighbor vertex, the cell vertex might collide with this neighbor vertex.

See Fig. 4.8(a). Assume we are analyzing cell vertex v_1 of the expanding site C_1. As site C_1 expands, v_1 moves along its radiating edge e_1 towards neighbor vertex v_2 (see Fig. 4.8(b)). No topology changes occur until v_1 collides with v_2. At the collision point, vertices v_1 and v_2 merge into a single cell vertex and radiating edge e_1 disappears (see Fig. 4.8(c)). After the collision, the new cell vertex splits into two cell vertices, which move along each of the newly adjacent radiating edges, generating a new cell edge e_2 (see Fig. 4.8(d)). This process is called *edge flipping* because it locally flips an old Voronoi edge (e_1) to a new one (e_2), with no other topology changes elsewhere.

We can perform a simple check for the collision between cell vertices and corresponding neighboring vertices by checking if the expanding site collides with the corresponding circumcircle of the neighbor vertex. In Fig. 4.8, neighbor vertex v_2 is the center of the circumcircle of its three corresponding sites C_2, C_3, and C_4. There is no topology change when the expanding site C_1 does not intersect the circumcircle (see Fig. 4.8(b)); two vertices degenerate to one when C_1 is tangent to the circumcircle (see Fig. 4.8(c), and an edge flip operation is needed when C_1 overlaps the circumcircle (See Fig. 4.8(d)).

Fig. 4.9 shows a special case where the neighbor vertex is in non-general position (i.e. the vertex is shared by more than three sites). In this case, after the cell vertex collides with this neighbor vertex, new cell vertices are generated on each of the radiating edges from the original neighbor vertex, and new cell edges are generated between these vertices, proceeding in clockwise or counter-clockwise order.

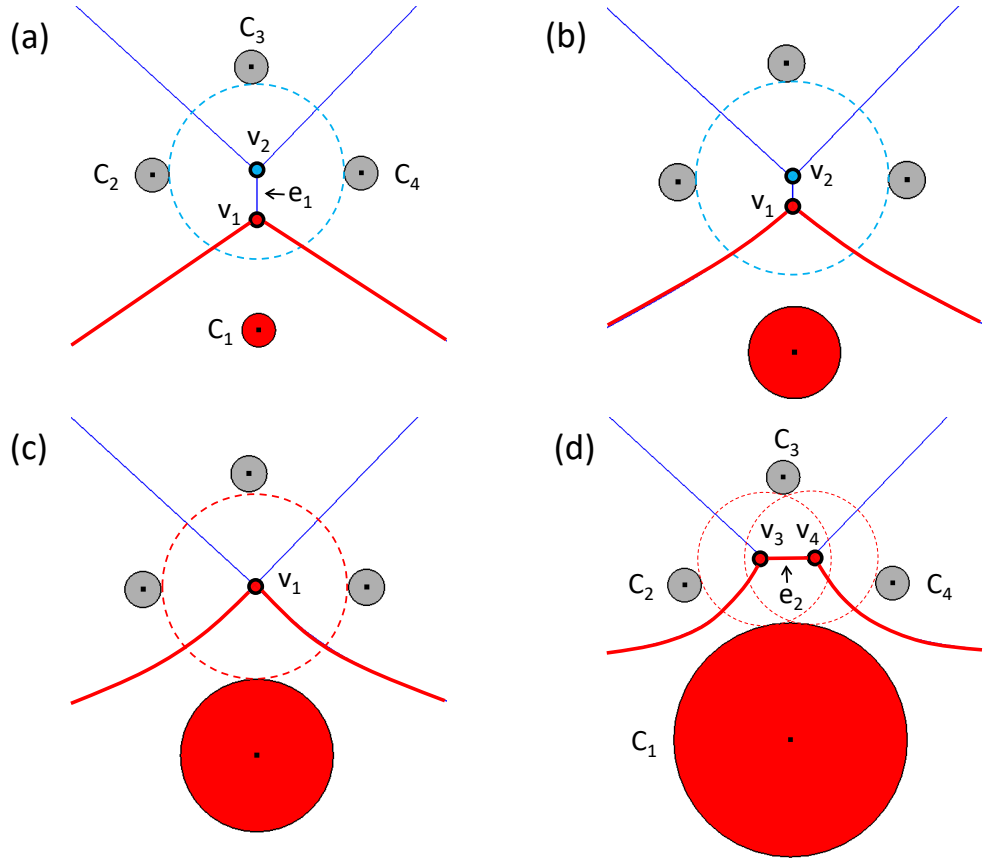

Figure 4.8: During expansion of a special site (red), the cell vertex v_1 collides with neighbor vertex v_2, causing a topology change.

After each of the topology change events described above, new vertices are generated. We iteratively check each newly generated vertex along its radiating edge, until no further topology changes occur.

Radiating edge extends to infinity

See Fig.4.10. When a cell vertex v_1 has a radiating edge e_1 that extends to infinity, it moves along this radiating edge. During the expansion of the corresponding cell, the vertex v_1 may disappear at a critical point. At the critical point, the original radiating edge e_1 disappears along with the cell vertex v_1, and instead the two cell edges connected to this cell vertex (e_2 and e_3) become infinite at this end (see Fig. 4.10).

We detect this topology change by calculating the circumcircle of the expanding cell vertex's corresponding input sites. If the circumcircle exists, the vertex is still moving along

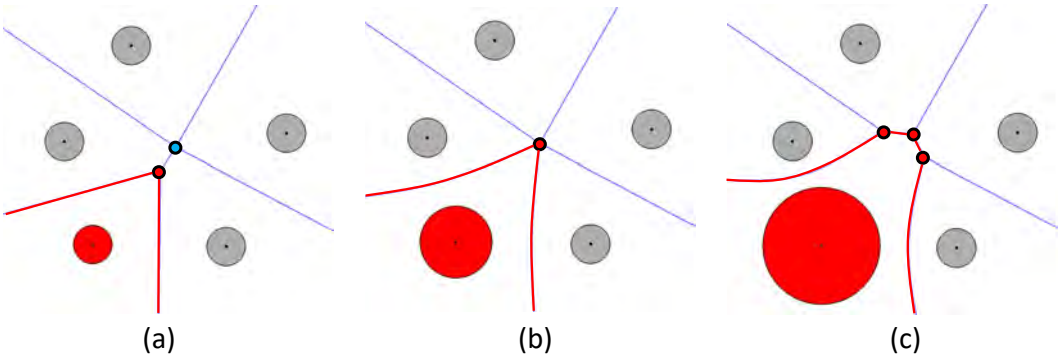

Figure 4.9: A special case: a neighbor vertex in non-general position.

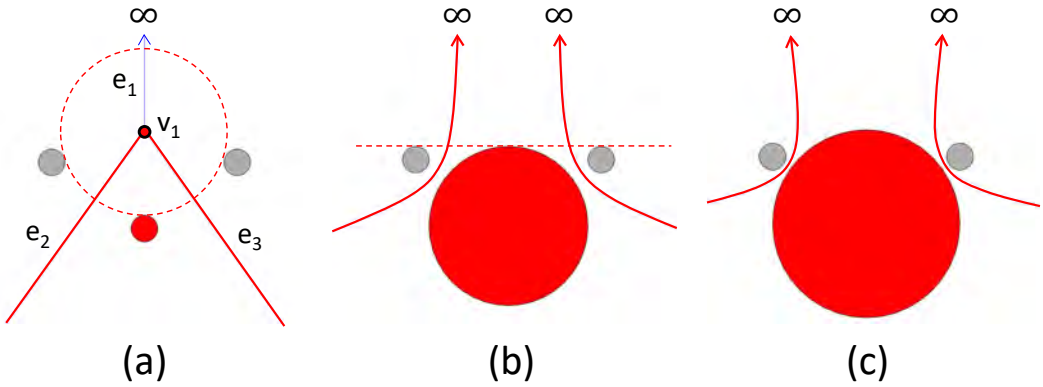

Figure 4.10: During the cell-expansion process, a cell vertex vanishes when its radiating edge extends to infinity

the radiating edge, and no topology change occurs (see Fig. 4.10(a)); the critical point occurs when all three corresponding inputs are tangent to a line (see Fig. 4.10(b)). Starting from this critical point, no circumcircles exist for the corresponding inputs, and the topology of the Voronoi diagram must be updated (see Fig. 4.10(b)(c)).

4.4.4 Overall cell-expansion process

Fig. 4.11 summarize of our cell-expansion process. The input is the ordinary Voronoi diagram, which is then processed in two levels of iteration in our method. We iterate over the special sites to expand their corresponding Voronoi cells. For each expanding cell, we first check for topology changes in its infinite edges and then perform another level of iteration to detect other topology changes by tracing each cell vertex along its corresponding radiating

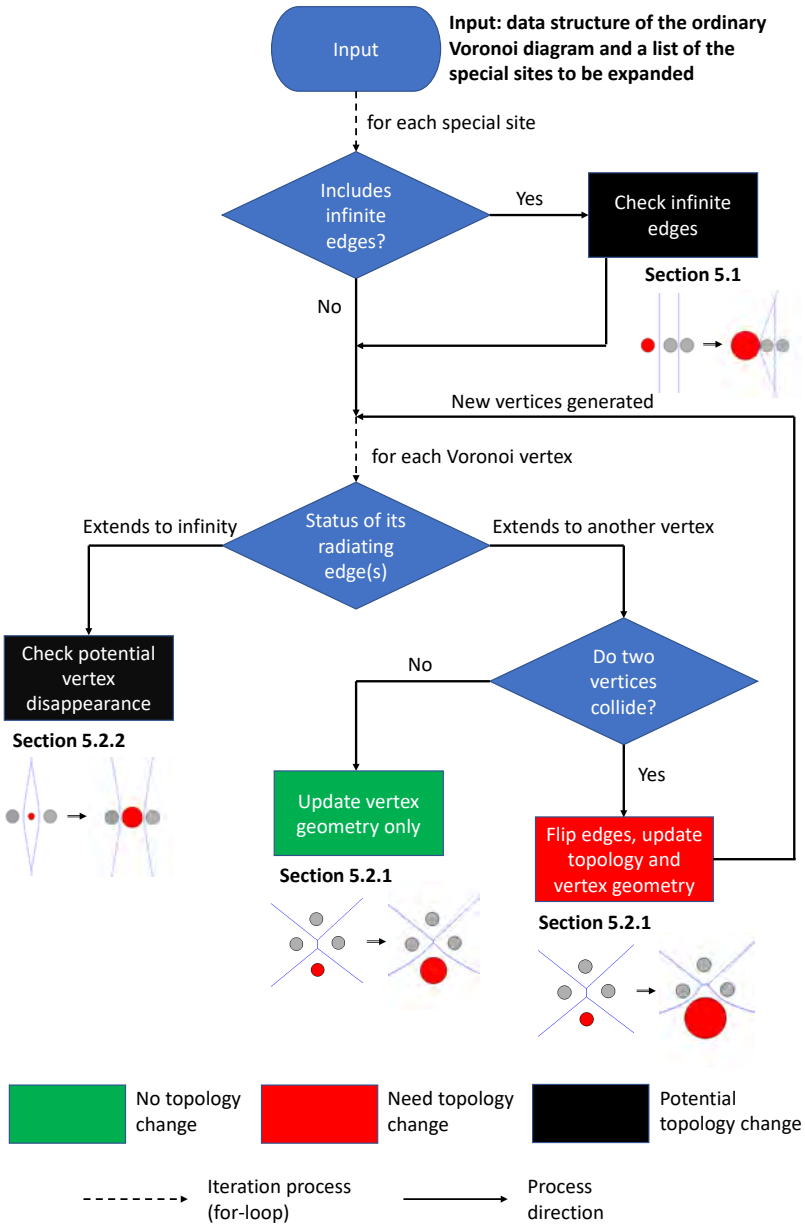

Figure 4.11: Summary of the cell-expansion process

edge(s).

During the process, we gradually update the topology and vertex geometry of the input Voronoi diagram until all special sites have been expanded. We do not calculate the curved edge geometry because it is not used in computing the α-shape in subsequent steps.

The cell expansion method works for both circle and ellipse inputs. Detecting topology

changes is based on detecting the intersections between the expanding object and each of the corresponding circumcircles formed by its neighboring vertices.

4.5 Construction of the α-shape

4.5.1 Determining the refined region

After calculating the Voronoi diagram, we directly obtain the dual triangulation. As Fig. 4.14(a) shows, at the boundary of the dual triangulation, skinny triangles can occur. In a later step of constructing the α-shape (Section 4.5.3), we will determine if the α probe can be placed in a dual triangulation cell by comparing the size of its corresponding circumcircle with the α value. In that step, such a boundary skinny cell will be falsely detected as a resin-rich area because the size of its circumcircle does not indicate the actual size of the empty space between the three corresponding input objects. To avoid this miscalculation, we must cull these *boundary-artifact* skinny cells from the dual triangulation. After this culling process, we determine "resin-rich areas" on the remaining dual triangulation area (see Section 4.5.3).

In the dual triangulation, for any triangle having at least one edge at the boundary, we determine if it is a boundary-artifact triangle by checking if its corresponding circumcircle entirely stays outside this triangle. If so, the size of the circumcircle depends more on the boundary geometry than the actual empty space of the boundary-artifact triangle (see Fig.4.12). We then cull all such boundary-artifact cells from the dual triangulation. We call the remaining part of the dual triangulation the *refined dual triangulation* (see Fig. 4.14(b)), and its corresponding region the *refined region*. The region outside the refined region is the boundary region (Fig. 4.14(c)), which is not considered in the following process.

Another potential way to handle Voronoi vertices outside the image is to compute all the intersection points of the Voronoi edges with the boundary of the image, and treat those intersection points as if they were Voronoi vertices for the purpose of identifying resin-rich areas: we would test whether circular probes centered at those boundary points intersect any fibers or not. We would still discard Voronoi vertices outside the image.

4.5.2 Determining the threshold for α values

In the microscope image, if all fibers are evenly distributed, there can be no resin-rich areas. See Fig. 4.13(a). In this optimal situation, the fibers are evenly distributed in a hexagonal lattice, with the plane tiled by a triangular pattern with vertices located at the centers of the three neighboring fiber cross-sections. The largest possible probe (with radius α_{thresh}) that can be placed into this pattern is the circle centered at the center of the triangle, tangent to all of the three fiber cross-sections (see Fig. 4.13(b)). Any geometry change to the fibers will cause what could be considered resin-rich areas, where probes with radii larger than α_{thresh} can be inserted. Thus only probes with radii larger than the threshold probe size (α_{thresh}) could indicate resin-rich areas.

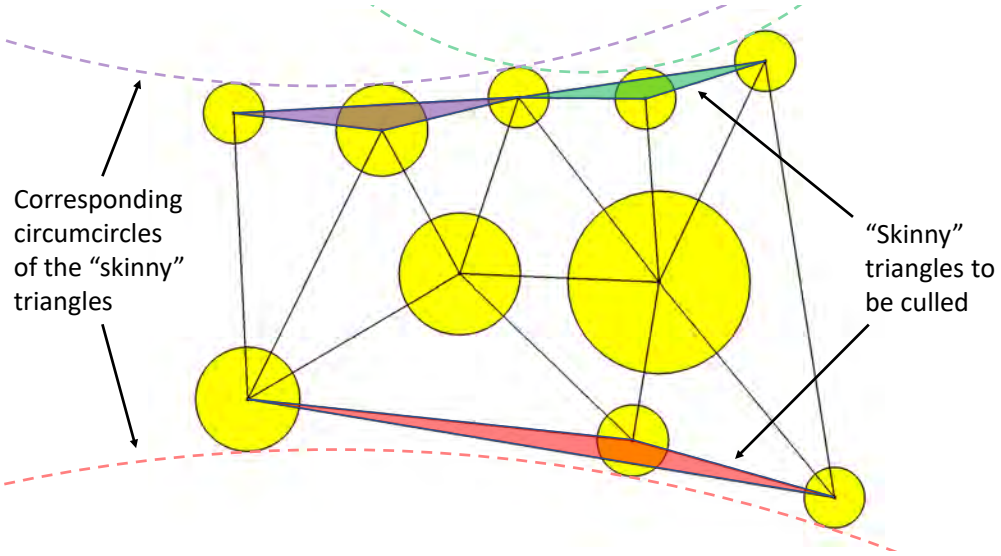

Figure 4.12: Boundary-artifact triangles and their corresponding circumcircles.

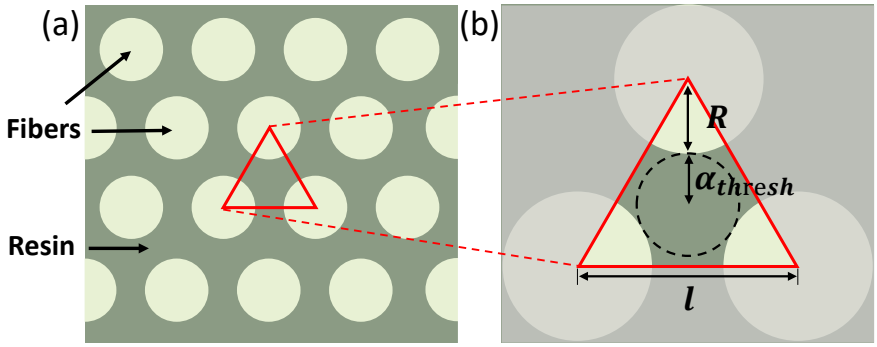

Figure 4.13: Geometric relation between the threshold α value α_{thresh} and the radius of the fiber cross-sections R.

This (α_{thresh}) is thus a function of the fiber volume fraction v_f (the ratio of the volume of fibers to the total volume) as well as the fiber radius R. Analyzing the tiled triangular geometry (see Fig. 4.13(b)) to find the ratio of the fiber cross-section areas to the whole cross-section area in this ideal case, we set it equal to the actual v_f for the manufacturing process and solve for α_{thresh}. Letting ℓ denote the side length of the equilateral triangle, we have:

$$v_f = \frac{\pi R^2}{\sin(60^\circ)\ell^2} \, . \tag{4.1}$$

For any equilateral triangle, the distance of its center from its vertices satisfies:

$$R + \alpha_{\text{thresh}} = \frac{\sqrt{3}}{3}\ell \ . \tag{4.2}$$

Substituting for ℓ in Eq. 4.2 by an expression in terms of R and v_f from Eq. 4.1, we can derive the following relation:

$$\alpha_{\text{thresh}} = \left(\frac{\sqrt{3}}{3} \sqrt{\frac{\pi}{\sin(60^\circ) v_f}} - 1 \right) R \ . \tag{4.3}$$

From the above equation, if the printed composite has, for example, nominal fiber volume fractions of 50%, 60%, or 70%, the threshold α value α_{thresh} is $0.555R$, $0.420R$, or $0.314R$ respectively. Choice of α_{thresh} in different applications is further discussed in Section 4.6.

4.5.3 Constructing the α-shape in the refined region

In [29], Kim et al. described a process for constructing the α-complex from the dual triangulation of circle inputs. We extend this idea to construct the α-complex of circles and ellipses from their refined dual triangulation. The process is has two parts.

Firstly, for each edge in the refined dual triangulation, we check if the probe can traverse it by comparing the probe's diameter (2α) with the shortest length between the edge's corresponding objects (see Fig. 4.15). For a dual triangulation edge between circles, the shortest length equals the edge length (distance between circle centers) minus the radii of the two corresponding circles; for a dual triangulation edge between an ellipse and another circle or ellipse, the shortest length no longer has a simple relation to its edge length. We apply the method proposed by Zheng et al. [54] to calculate the shortest distance between two arbitrary ellipses, or a circle and an arbitrary ellipse. If the edge's corresponding shortest length is greater than 2α, the probe is able to freely traverse such an edge without colliding with either of the objects, so we keep this edge in the α-complex; if not, the probe is unable to traverse the edge, so we remove it from the α-complex.

Secondly, for each cell in the refined dual triangulation (connecting a triplet of circle centers), we check if the probe can be placed into it by comparing the diameter of the cell's corresponding circumcircle with 2α. This is the circumcircle we calculated when constructing the Voronoi diagram, the largest circle that can possibly be placed between the three corresponding inputs (see Figures 2.1 and 2.3). If the circumcircle's diameter is smaller than 2α, the probe can not be placed in this cell, and we keep this dual triangulation cell in the α-complex; otherwise, the probe can be placed in the cell, and we remove the cell from the α-complex.

The α-complex (Fig. 4.14(d)) is a subset of the dual triangulation; it may contain dangling edges and interior voids. The above processes for checking edges and cells are separate: removing a cell does not necessarily mean its three corresponding edges will also be removed from the α-complex.

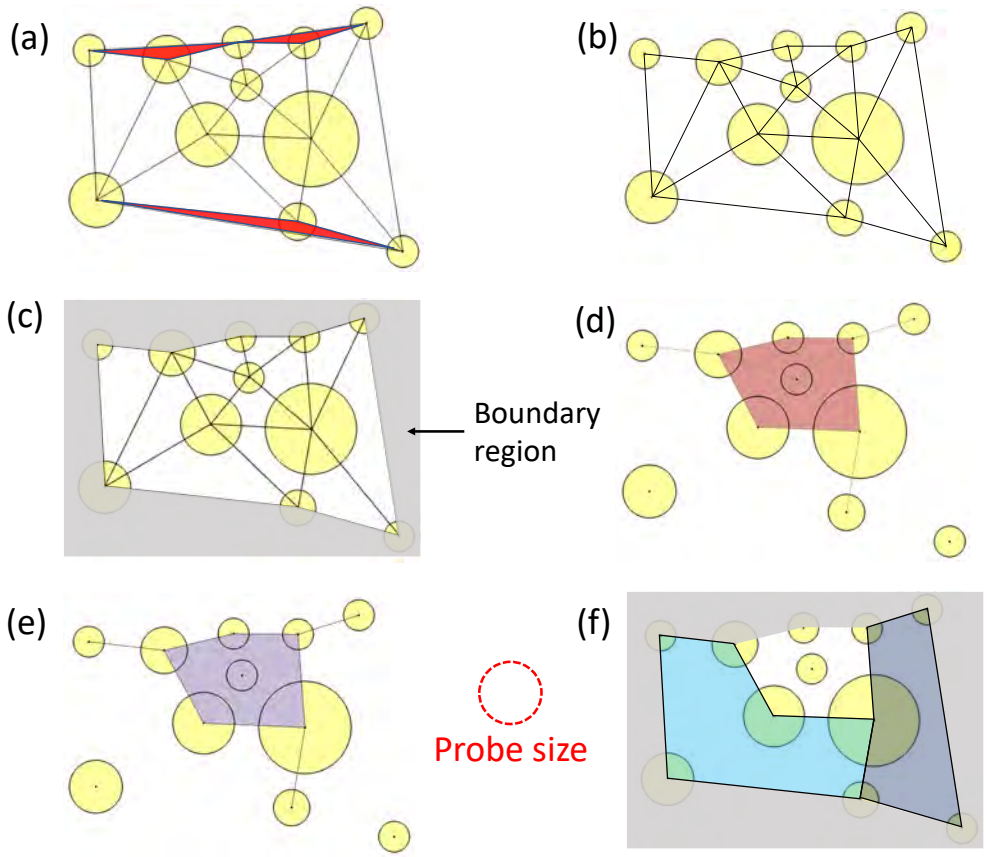

Figure 4.14: Determining resin-rich areas from the dual triangulation. (a) dual triangulation, with boundary-artifact cells rendered in red, (b) refined dual triangulation, (c) boundary region, which is the complement region of the refined dual triangulation, (d) α-complex, (e) α-shape, (f) detected resin-rich areas.

From the α-complex, we merge all neighboring cells, and remove the edges between merged cells. The new structure is the α-shape (see Fig. 4.14(e)), indicating areas within which fibers are sufficiently dense that no probes could be placed there.

To find the resin-rich areas, in the refined region, we take the complement of the α-shape. If a complementary area is fully divided by the dangling edges in the α-shape, we treat its sub-areas as separate resin-rich areas (see Fig. 4.14(f)). It is necessary to distinguish such sub-areas because they do not form a fully connected region in which fibers are deficient. For example, see Fig. 4.16: although fibers are deficient inside each of the two detected sub-areas, there are enough closely spaced fibers at the boundary where they meet to increase its local strength. Since the resin-rich areas are not continuous across their locally reinforced contacting boundary, considering them as separate sub-areas helps in subsequent quality analysis and failure prediction processes.

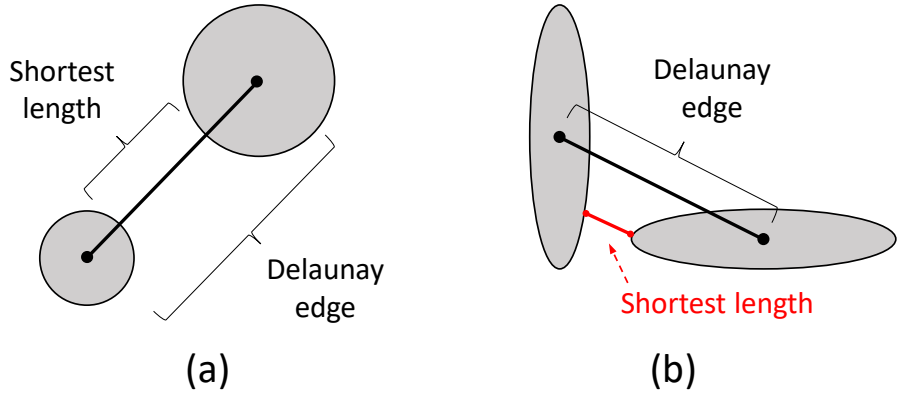

Figure 4.15: The shortest length between: (a) two circles; (b) two ellipses.

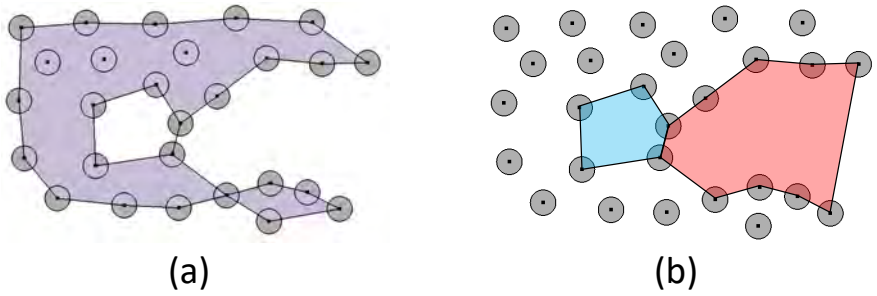

Figure 4.16: An example of the resin-rich area divided by a dangling edge: (a) α-shape; (b) two detected resin-rich areas.

4.6 Results

We have tested our algorithm on real-world microscope images from 3D printed FRP parts. Each cross-sectional microscope image usually contains 100,000 to 1,000,000 fiber cross-sections. In our images, the mode of the radii of fiber cross-sections is around 7 pixels. Because of the inherent variability in microscopy and circle/ellipse detection processes, it is common to have a ± 1 pixel deviation in the detected circle radii or ellipse minor axes (for a misaligned fiber, the minor axis appearing in its cross-section equals the original fiber radius). Therefore we use $\pm 15\,\%$ ($\pm\lceil 1/7 \rceil$) as our acceptable tolerance: if a fiber cross-section does not deviate by more than $15\,\%$ from the mode of the detected radii R_{mode}, we consider it to have radius R_{mode} just like most other fiber cross-sections.

The proportion of expanded and misaligned fibers is highly dependent on the quality of the 3D printing process: poorer manufacturing leads to more expanded and misaligned fibers. In our test images, we found that no more than 1% of the fiber cross sections were

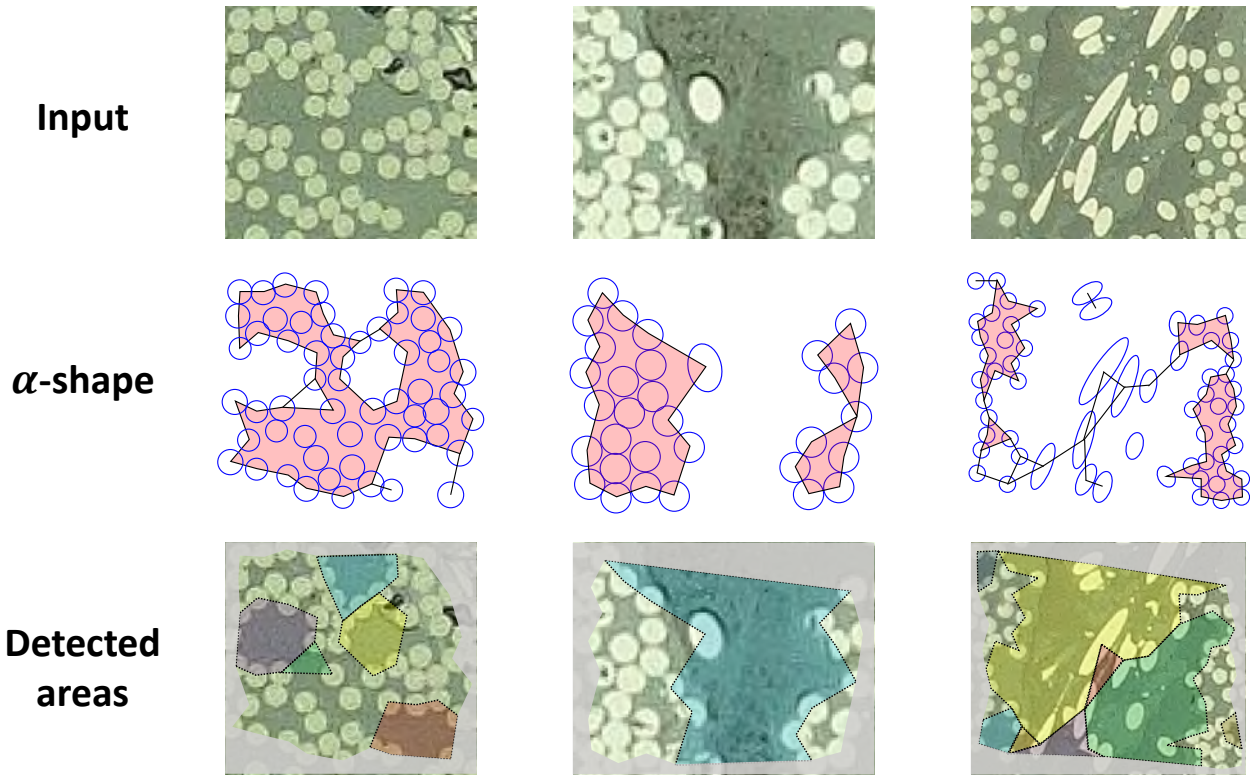

Figure 4.17: Experimental results of three real-world microscope images. Separate detected areas are distinguished with different colors; grayed out areas are boundary regions.

larger size (expanded) circles ($radii > 1.15R_{\text{mode}}$) and no more than 0.008% were ellipses. No shrunken fibers ($radii < 0.85R_{\text{mode}}$) were found. However, in the circle/ellipse detection, it is possible that broken fibers will appear as small-radius shrunken fibers. In that case, we can just treat them as absent because the strength of the broken fibers are mostly lost; the areas where they appear remain "resin-rich."

Our algorithm robustly handled all the 30 real-world examples we tested, whether the fibers are circles or ellipses, in general position or not (see Fig. 4.17 and 4.18). To better demonstrate our results, in Fig. 4.17, we show our implementations on small cropped portions of microscope images with different levels of ellipse content ratio. The probe radius α is defined as the mode of the radius of the detected fibers ($\alpha = R_{mode}$) in each test case.

Different choices of α result in different detected areas (see Fig. 4.18). Higher α values keep significant fiber-deficient areas and ignore tiny ones, useful when we only need to locate large defects such as inter-layer or inter-strip fiber-deficient areas. Lower α detects both significant and tiny fiber-deficient areas, which is preferred when we want to compile thorough statistics of fiber distribution in the part. For high-quality industrial FRP composite parts (typically 50%–70% nominal fiber volume fraction), depending on the application, α values

ranging from R_{mode} to $3R_{\text{mode}}$ provide good inspection results.

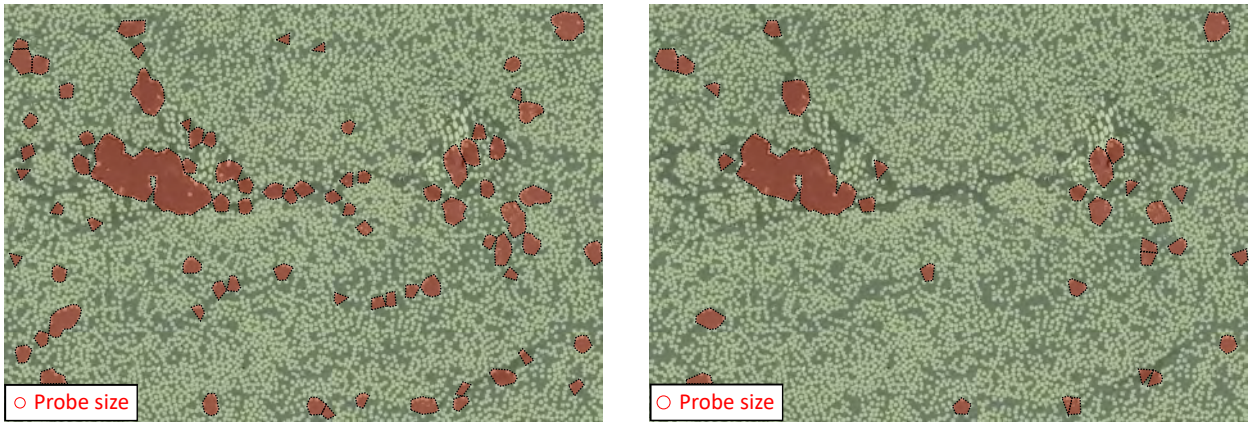

Figure 4.18: Effect of changing α. Left: 79 detected fiber-deficient areas when $\alpha = 2R_{\text{mode}}$. Right: 31 detected areas when $\alpha = 2.5R_{\text{mode}}$.

To perform a complexity analysis, let n be the number of input sites (detected fiber cross-sections, including expanded and misaligned ones). It takes $\mathcal{O}(n \log n)$ time and to construct the ordinary Voronoi diagram of points and thus of the centers of detected fiber cross-sections. The cell expansion process of each large circle or ellipse takes time linear in the number of neighboring sites of the expanded cell. Since there might be $(n-1)$ large circles and ellipses, and each expanded cell may interact with all the other $(n-1)$ sites, the cell-expansion process takes $\mathcal{O}(n^2)$ time in the worst case. In most cases, the cell-expansion process takes $\mathcal{O}(n)$ time because in general Voronoi cells have few, $\mathcal{O}(1)$, neighboring sites. Constructing the dual triangulation from the Voronoi diagram only takes $\mathcal{O}(1)$ time because the dual is already captured in the Triangle software's data structure. From the dual triangulation, we compare the size / shape of each of the $\mathcal{O}(n)$ edges and cells to the size of the α-probe, and preserve those that are not traversable by the α-probe as the α-shape. Therefore, the total time complexity of our algorithm is $\mathcal{O}(n \log n)$ for typical real-world inputs, even if $\mathcal{O}(n^2)$ in the worst case.

We have implemented our algorithm in C++, and tested our code on a PC with an Intel Core i7-9700K CPU with 16GB RAM. To test the efficiency of our implementation, we randomly selected real-world microscope images and cropped them to give smaller images containing different numbers of fibers. Run times (exclusive of the fiber detection process) for different cropped images are shown in Table 4.1. We can observe that, although the calculation complexity depends on the number of expanded and misaligned fibers, the run time is roughly linear with regard to the number of detected fibers in these real-world inputs. However, when the input includes a large number of misaligned fibers, the algorithm takes significantly more time, a limitation of our approach.

Table 4.1: Run time (excluding circle/ellipse detection) for real-world inputs with different numbers of detected fibers.

Number of detected fibers	Number of expanded fibers	Number of misaligned fibers	Run time (s)
10,087	85	0	0.25
50,048	40	5	2.17
100,040	92	3	3.12
500,156	1418	20	25.67
1,000,144	4482	144	68.63

Table 4.2: Run time under different number of large circles within input containing 100,040 detected fibers.

Number of large circles	Number of ellipses	Run time (s)
0	0	2.51
1,000	0	2.77
2,000	0	3.01
5,000	0	3.70
10,000	0	4.76

Table 4.3: Run time with different numbers of ellipses within input containing 100,040 detected fibers.

Number of large circles	Number of ellipses	Run time (s)
0	0	2.51
0	10	3.84
0	20	4.28
0	50	6.78
0	100	10.94
0	500	53.96

To test the sensitivity of our algorithm to the number of expanded or misaligned fibers, we artificially increased their content ratios by randomly picking regular detected circles and reshaping them into large circles or ellipses. For an example with 100,040 detected fibers, the algorithm's run time for different numbers of artificially expanded circles and ellipses are shown in Tables 4.2 and 4.3 respectively. Compared to the number of large circles, the number of ellipses has more impact on the total run time, due to the more complicated calculation of circumcircles and shortest distances for ellipses.

4.7 Conclusions

This chapter presents a novel algorithm to automatically detect resin-rich areas from microscope images of 3D printed FRP parts. It successfully handles real-world microscope images containing more than 1,000,000 fiber cross sections in 70 seconds, whether the fibers are aligned or misaligned, in general position or non-general position. We exploit the particular characteristics of fibers in 3D printed FRP parts in our design to considerably improve the efficiency over general-purpose geometric construction algorithms. Although the computation time is sensitive to the unpredictable number of misaligned fibers, in our real-world examples, it shows a roughly linear relationship to the number of fiber cross sections.

Our algorithm works not only for circles and ellipse, but is also applicable to all convex shapes. Following the same process, one could extend this algorithm to construct Voronoi diagrams and α-shapes of other convex shapes by calculating circumcircles and shortest distances among them.

Chapter 5

Voronoi Diagram of Spheres

5.1 Introduction

While detecting resin-rich areas using the Voronoi approach (discussed in Chapter 4), note that the most computationally expensive step is the exact computation of circumcircles in the construction of Voronoi diagrams, taking about 70% of the total computation time. For example, for a real-world high-resolution (18,000*10,000 pixels) microscope image containing about 1.15 million fiber cross-sections, it takes 74.48 seconds for the Voronoi approach to detect resin-rich areas, in which 52.79 seconds are the exact computation of circumcircles (and 61.33 seconds for the whole Voronoi construction). Considering this situation, a potential approach to further increase the efficiency of the resin-rich area detection process is to use sample-based techniques to construct the Voronoi diagram, which could avoid complex exact computations.

In Hu et al. [50], we proposed an algorithm to calculate the geometry information of Voronoi vertices and Voronoi face sample points. The algorithm is a sample-based approach that calculates sample points on Voronoi faces by taking the lower envelope of the intersections of rays from each base sphere through its corresponding bisectors. It was able to find Voronoi vertices of both general and non-general position (degenerate-case) inputs by searching for patterns of neighboring sample points that indicate the presence of Voronoi vertices and using numerical iteration to calculate the vertex locations.

We would like to test if the resin-rich area detection process could be more efficient by substituting (the 2D version of) this sample-based Voronoi construction algorithm for our exact computation (cell expansion) process described in Chapter 4. However, this sample-based algorithm only calculates the geometry information of Voronoi vertices; the Voronoi edge topology information is not calculated. Without calculating the topology information, we are not able to integrate this algorithm into our resin-rich area detection process because the dual triangulation can not be determined.

Thus, in this chapter, we develop a follow-on algorithm to calculate Voronoi edges under both general or non-general input positions (including disconnected Voronoi edges and

degenerate cases). Combining our output with the prior geometry output from [50], we show how to build the full Voronoi diagram for 3D spheres with inputs under any condition (Fig. 5.1). We also test the efficiency of our resin-rich area detection process by applying the 2D version of this sample-based method in the Voronoi construction step.

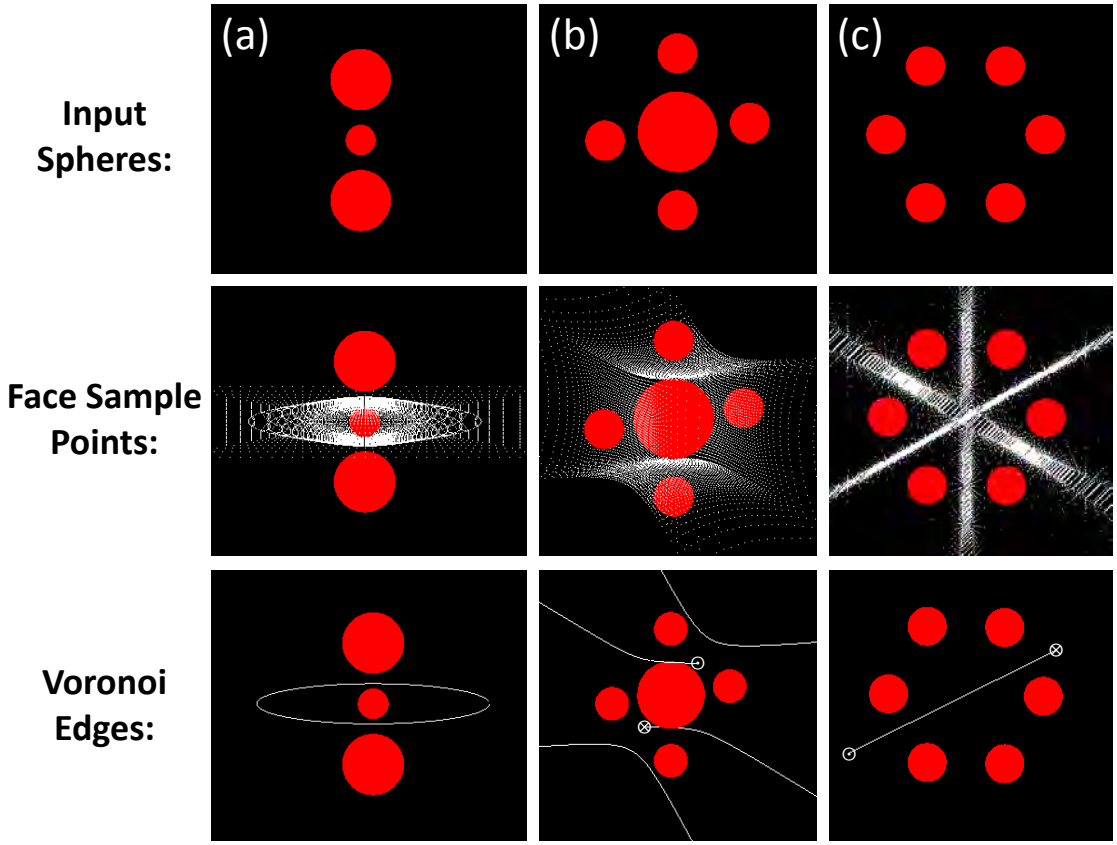

Figure 5.1: Example results showing correctly identified 8 Voronoi edge topology for challenging special cases (all figures are 2D rendering of 3D scenes): (a) A self-connected ring-shaped Voronoi edge is identified (the centers of these three input spheres lie on the same line); (b) Four infinite Voronoi edges (both ends extending to infinity) are identified for this case where the centers of the five input spheres lie on the same plane; (c) An infinite Voronoi edge (with both ends extending to infinity) is identified for this case where the centers of six input spheres lie on the same plane. The Voronoi edge passes through the center of the ring of six spheres and is perpendicular to their plane. Symbol "$\odot$" represents Voronoi edge shooting outwards to infinity (coming towards the reader); symbol "$\otimes$" represents Voronoi edge shooting inwards to infinity (away from the reader).

5.2 Terminology and definitions

Following [55], the Voronoi diagram for a set of spheres in 3-dimensional space is defined as:

Definition 1: Given a set of spheres $S_0, S_1, \ldots, S_n$ in $\mathbb{R}^3$, the Voronoi cell (VC) of sphere S_i, denoted the "base sphere," is the set of all points closer to S_i than to $S_j, \forall j \neq i$. The Voronoi diagram (VD) is then the union of the Voronoi cells of all (n+1) spheres.

The distance between a point $P = (x, y, z)$ and a sphere S with center (C_x, C_y, C_z) and radius R is defined by the equation:

$$dist(P, S) = \sqrt{(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2} - R. \tag{5.1}$$

Definition 2: The union of all points that are equidistant from spheres S_i and S_j is called the bisector $B_{i,j}$ of the two spheres. S_i, S_j are called the generating spheres of the bisector $B_{i,j}$.

Any point $P = (x, y, z)$ located on the bisector surface between two spheres S_1 (center $(C_{x_1}, C_{y_1}, C_{z_1})$ and radius R_1) and S_2 (center $(C_{x_2}, C_{y_2}, C_{z_2})$ and radius R_2) satisfies the following equation:

$$\begin{aligned} &\sqrt{(x - C_{x_1})^2 + (y - C_{y_1})^2 + (z - C_{z_1})^2} - R_1 \\ &= \sqrt{(x - C_{x_2})^2 + (y - C_{y_2})^2 + (z - C_{z_2})^2} - R_2. \end{aligned} \tag{5.2}$$

The bisector is a plane for two generating spheres with the same radii; for two generating spheres with different radii the bisector is a hyperbolic surface [50, 56]. A Voronoi face is the subset of a bisector that is closer to its generating spheres than to any other spheres.

Within a Voronoi cell, Voronoi edges are the intersection between two of its Voronoi faces. In general, Voronoi vertices are the intersection among three Voronoi faces; such a vertex is determined by the four spheres to which it is equidistant (the base sphere and three other spheres corresponding to each of the Voronoi faces).

A base sphere is a generating sphere of a Voronoi face/edge/vertex if such a Voronoi face/edge/vertex appears in the Voronoi cell corresponding to the base sphere. Typically, a Voronoi face/edge/vertex has 2/3/4 generating spheres, respectively. If there are more generating spheres than this general case, they are said to be not in "general position."

5.3 Prior algorithm to calculate Voronoi vertices

We briefly summarize our prior algorithm [50] as follows:

1. Determine the implicit quadratic surface equations, derived from Eqn. 5.2, for the bisectors between the base sphere and all the other input spheres (input spheres can intersect but not completely contain another sphere).

2. From each base sphere, the algorithm creates an axis-aligned bounding cube, and uniformly subdivides each of the six faces to a parameterized domain expressed in

variables u and v (Fig. 5.2). From the center of the base sphere, the algorithm shoots sampling rays through each (u, v) sample point on the bounding-box surface into space.

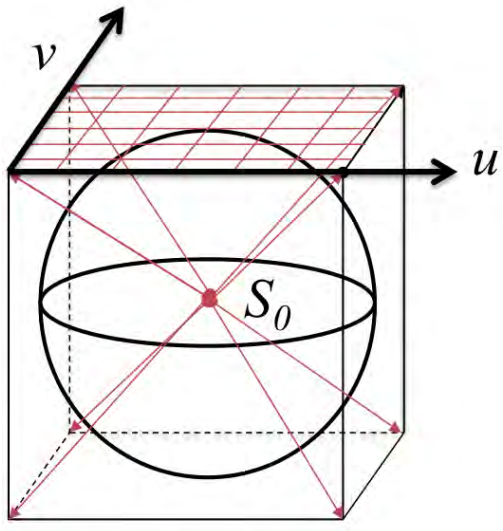

Figure 5.2: Mapping sphere to six u-v parametric surfaces on the bounding cube; uniform parametric sampling of top surface shown [50].

3. Compute the intersection of each base sphere ray with all the corresponding bisectors, and take the lower envelope of all the intersections (i.e. only keep the nearest intersection for each ray) to obtain the sample points on the Voronoi faces of the Voronoi cell for this base sphere.

4. The algorithm color-codes each sample point of the Voronoi cell for the base sphere on the u-v parametric domain based on its corresponding bisector found in step 3. It uses a marching approach to locate the neighborhood of Voronoi vertices by checking each group of four neighboring sample points on the bounding cube, called a "grid-cell." Each 3-color and 4-color grid-cell indicates the appearance of three or more Voronoi faces in this neighborhood. Recall that Voronoi vertices are the intersection among three Voronoi faces in general, so 3-color and 4-color grid-cells indicate the existence of Voronoi vertices.

   Fig. 5.3 shows the correspondence between sample points in geometric space and the u-v parametric domain.

5. For each 3-color grid-cell, take the average of the location in 3D space of the face sample points as the starting point for iteration, then use the Newton-Raphson method to find the actual vertex location (within a user-defined tolerance) that satisfies the three corresponding implicit bisector equations.

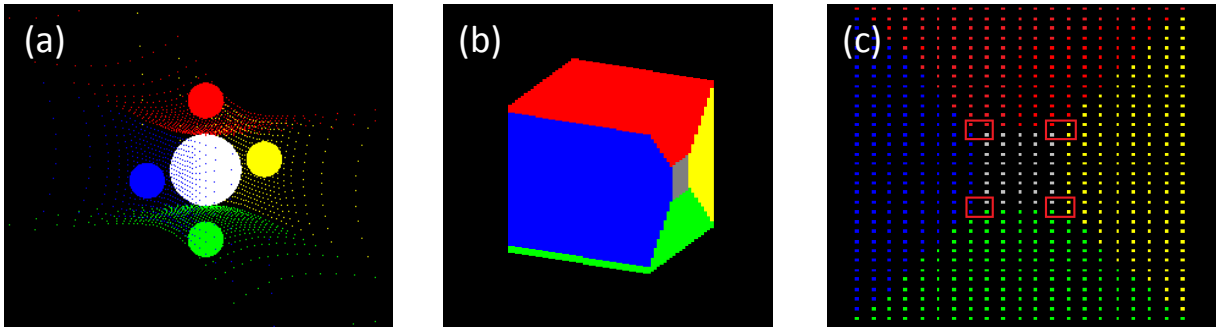

Figure 5.3: (a) Sample points on Voronoi faces for white base sphere with four spheres of the same size evenly spaced around it, all five with co-planar centers; (b) corresponding color map of u-v domains on bounding cube with gray representing sample rays that go to infinity; (c) sample point grid on one face of the bounding cube with 3-color grid-cells indicated by boxes [50].

If the sampling density is insufficient, special cases of singular Jacobian and 4-color grid-cells would occur, indicating that Voronoi vertices cannot be calculated in those grid-cells. The algorithm uniformly subdivides such grid-cells into four new sub-grid-cells, repeating steps 1-5 for the newly generated u-v points. New sub-cells shoot additional sample rays to the 3D space neighborhood corresponding to the original grid-cell, increasing the local sampling density to provide more information to calculate the Voronoi vertices.

6. The algorithm combines the results of the Voronoi cell calculated for each base sphere to form the full Voronoi diagram. Because each Voronoi vertex has multiple generating spheres (four for general position or more for non-general position), it should be found from all of the Voronoi cells corresponding to the generating spheres. When the sampling density is insufficient, some Voronoi vertices may not be found from all the Voronoi cells. For such "incompletely matched" vertices, from each corresponding base sphere whose Voronoi cell did not find it, the algorithm shoots a new ray from the center of the base sphere to the exact location of this point (the exact location calculated from the other Voronoi cells that found it). The intersection of this ray with the bounding-box surface is the corresponding u-v location of the vertex. Around this vertex's u-v location, the algorithm constructs a much smaller grid-cell (Fig. 5.4), repeating steps 1-5 for the four newly generated u-v points of the smaller grid-cell. After this targeted sampling around the vertex, the tiny grid-cell will typically have the 3-color patterns corresponding to its generating spheres.

Please refer to [50] for complete details of the prior algorithm.

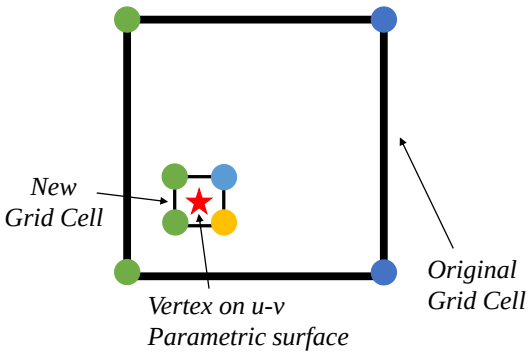

Figure 5.4: Construction of the new tiny grid-cell [50].

5.4 Updates in the construction of geometry information

In this section, we revisit some implementation details for the prior algorithm summarized above. We turn the process of creating tiny grid-cells of incompletely matched vertices into an iterative searching process (section 5.4.1), improving the robustness of the prior algorithm and establishing the relationship between the tiny grid-cell and its containing grid-cell. On the u-v domain, we track the neighboring information between grid-cells by adding pointers in the data structure, and update the information (pointers) during subdivision and the iterative searching process (section 5.4.2).

5.4.1 Iterative search for incompletely matched vertices

In step 6 of the prior algorithm, tiny grid-cells are created to check if the vertex actually exists in the Voronoi cells that did not find it [50]. If the tiny grid-cell is a 3-color or 4-color grid-cell, and its colors are consistent with those in the other cells that initiated the targeted search (Fig. 5.5(a)), the vertex exists. (Note that for a non-general position Voronoi vertex that has more than four corresponding colors (contributing spheres), if the three or four colors from the tiny grid-cell are a subset of those corresponding colors, it is also considered consistent with other contributing spheres.) If the colors are not consistent, the algorithm will recursively create even smaller grid-cells using the same reduction ratio, until a consistent 3-color or 4-color grid-cell is found or reaching a maximum depth of recursion.

The prior algorithm only calculated the Voronoi vertices' geometry, not their connectivity via Voronoi edges. To calculate the latter, we need to connect each of the four sample points on the tiny grid-cell to each of the corresponding four sample points on its original containing grid-cell. For example, as shown in Fig. 5.5(b), the blue sample point on the bottom-right corner of the tiny grid-cell is connected to the blue sample point on the bottom right corner

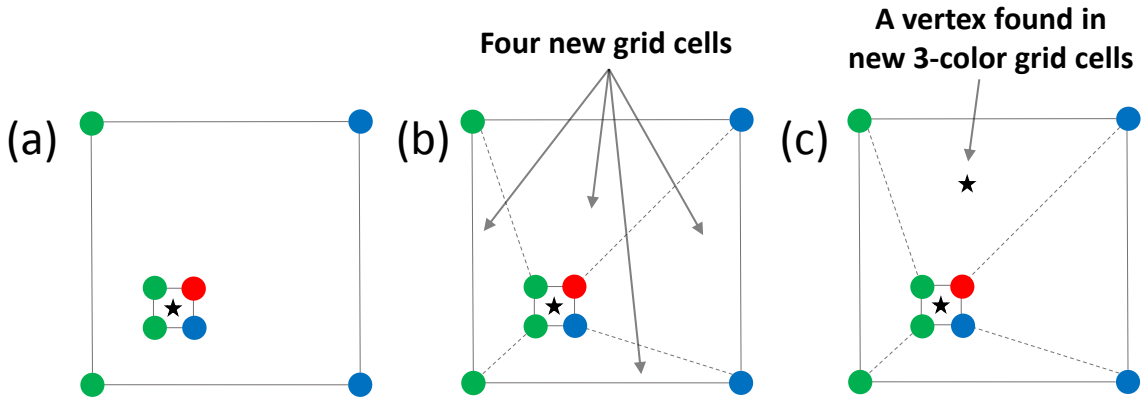

Figure 5.5: One iteration in the iterative search process for incompletely matched vertices.

of the original grid-cell.

By this process, we will create four new grid-cells (in addition to the tiny grid-cell). If any of these new grid-cells is 3-color or 4-color, it corresponds to a new vertex that has not been found in this Voronoi cell before (Fig. 5.5(c)). We calculate the position of this new vertex using numerical iteration and check that it appears in the Voronoi cells all of its other generating spheres (corresponding to the 3 colors of the grid-cell). If it is missing for any of the generating spheres, we repeat this process for this new incompletely matched vertex (in the u-v domain of any generating sphere that does not yet contain it in its Voronoi cell).

We repeat this process for each new incompletely matched vertex, until no more such vertices are found.

5.4.2 Create neighboring information of grid-cells

Just as in the u-v domain we call each group of four neighboring face sample points a "grid-cell," similarly we call each pair of neighboring face sample points a "grid-side." Each grid-cell has four grid-sides initially (if neighboring cells are subdivided, its sides will also be split whenever a new sample point is introduced in the middle of a side). A grid-side connecting a pair of sample points of the same color is called a homogeneous grid-side; a grid-side connecting a pair of sample points of different colors is called a heterogeneous grid-side. Each grid-side has two neighboring grid-cells that both contain this grid-side. If it is on the edge of the bounding cube, it is shared by two grid-cells on different u-v parametric surfaces (Fig. 5.6); these grid-cells are still neighbors because of sharing the same grid-side. Having this kind of neighborhood relationship allows traversing between different parametric surfaces on the same bounding cube. In the subdivision process, we may divide an original grid-cell into four sub-grid-cells (e.g. sub-grid-cells (6, 7, 8, and 9) in Fig. 5.7). We use sub-grid-cell 6 as an example; it shares small grid-sides e_1 and e_2 with original grid-cell 1 and 5

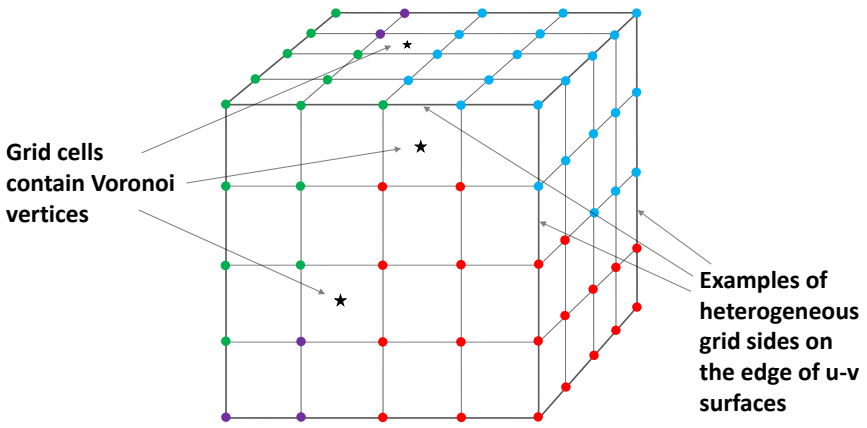

Figure 5.6: Color map on a bounding cube (sampling density 5*5).

respectively. In such situations, we still describe them as neighboring grid-cells: grid-cells 1 and 6 are neighbors by edge e_1, and grid-cells 5 and 6 are neighbors by edge e_2. Grid-cell 1 has five neighboring grid-cells (2, 3, 4, 6, and 7); Sub-grid-cell 6 has four neighboring grid-cells (1, 5, 7, and 9). As shown in Fig. 5.8, in the case of iterative targeted sampling,

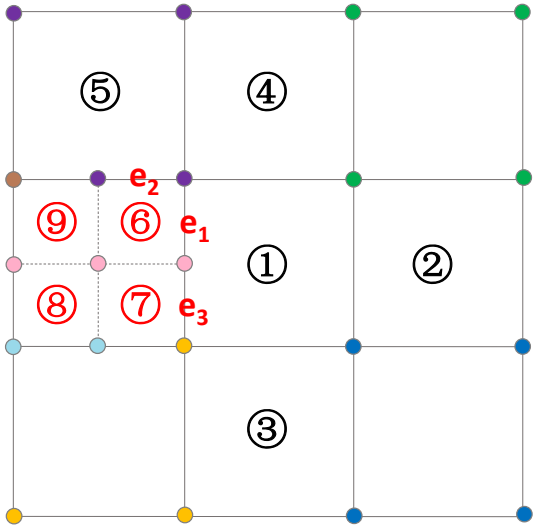

Figure 5.7: Neighboring information after subdivision.

we create a tiny grid-cell (6) around the u-v domain location of the vertex. By connecting the four corner points of the tiny grid-cell to the corresponding four corner points of the

original grid-cell, grid-cells (7, 8, 9, and 10) are created. Their neighboring information is still determined by shared edges.

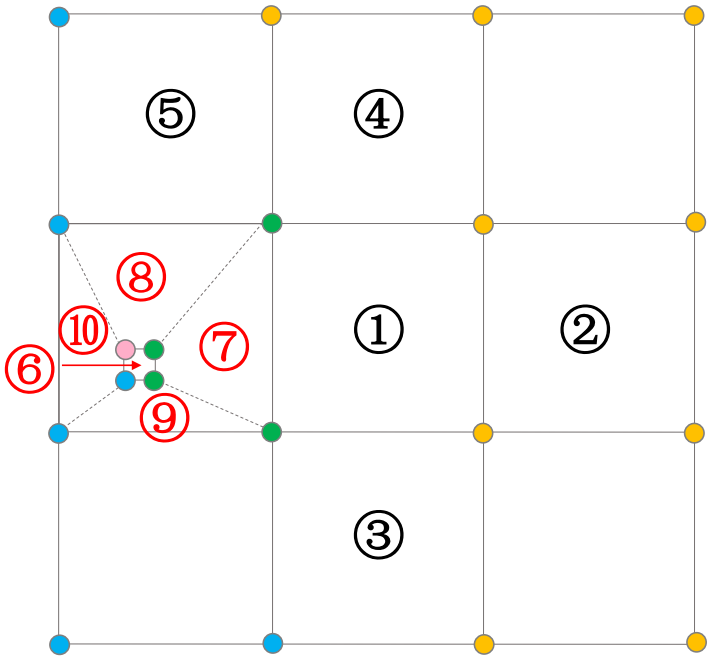

Figure 5.8: Neighboring information after targeted search.

5.5 Construction of edge topology information

In the previous sections, we described how to obtain the exact locations of the Voronoi vertices and their corresponding u-v grid-cells from each of the contributing base spheres. To determine the connectivity among the Voronoi vertices, we detect Voronoi edges by exploring the connectivity of grid-cells on the colored bounding cubes (u-v parametric surfaces).

Within a Voronoi cell, each Voronoi edge is the intersection between two of its Voronoi faces, so neighboring pairs of u-v sample points of different colors (heterogeneous sides of grid-cells) indicate the existence of Voronoi edges in its corresponding neighborhood in 3D space. Each heterogeneous grid-side indicates the presence of a particular Voronoi edge generated by the base sphere and two spheres corresponding to the two differently colored sample points. The pair of colors is called the "edge identifier" of its corresponding Voronoi edge in this Voronoi cell. For a particular Voronoi edge, we look for heterogeneous grid-sides with the same edge identifiers. If a grid-cell has two heterogeneous grid-sides with the same color pair (edge identifier), it indicates this particular edge enters this grid-cell from the neighboring grid-cell sharing one such side, and exits to the neighboring grid-cell sharing

the other such side. We call such grid-cells "through-grid-cells" of a particular Voronoi edge because the edge goes through those grid-cells.

As illustrated in Fig. 5.9, our algorithm premise is straightforward: tracing the paths of each Voronoi edge by following series of its "through-grid-cells." (Note that henceforth we are using the term "edge tracing" in this sample-space context; it has no relation to the "edge tracing algorithm" of Kim et al.) In a grid-cell containing a Voronoi vertex, each heterogeneous grid-side represents one particular Voronoi edge that exits into the neighboring grid-cell that shares this grid-side. Starting from each grid-cell that contains a Voronoi vertex, we trace the paths of each of its incident Voronoi edges along a sequence of through-grid-cells connected by grid-sides with the edge identifier associated with that Voronoi edge, until reaching another grid-cell also searching with the same edge identifier.

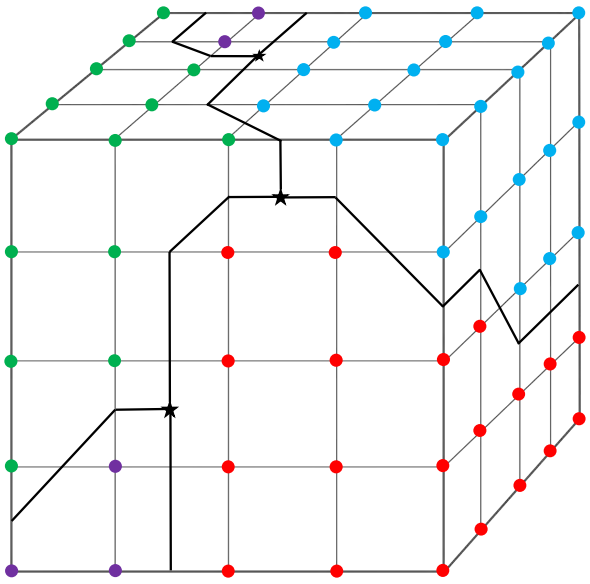

Figure 5.9: Voronoi edge topology on bounding cube (sampling density 5*5).

Our algorithm has four stages: preprocessing 2-color grid-cells with particular color patterns (Section 5.5.1), tracing "through grid-cells" on the bounding cube (Section 5.5.2), searching for isolated Voronoi edges (Section 5.5.3), and sorting of the Voronoi edges (Section 5.5.4).

We now describe the steps in detail.

5.5.1 Subdivision preprocessing of 2-color grid-cells

There are three possible configurations (and their inverses) of 2-color grid-cells as shown in Fig. 5.10.

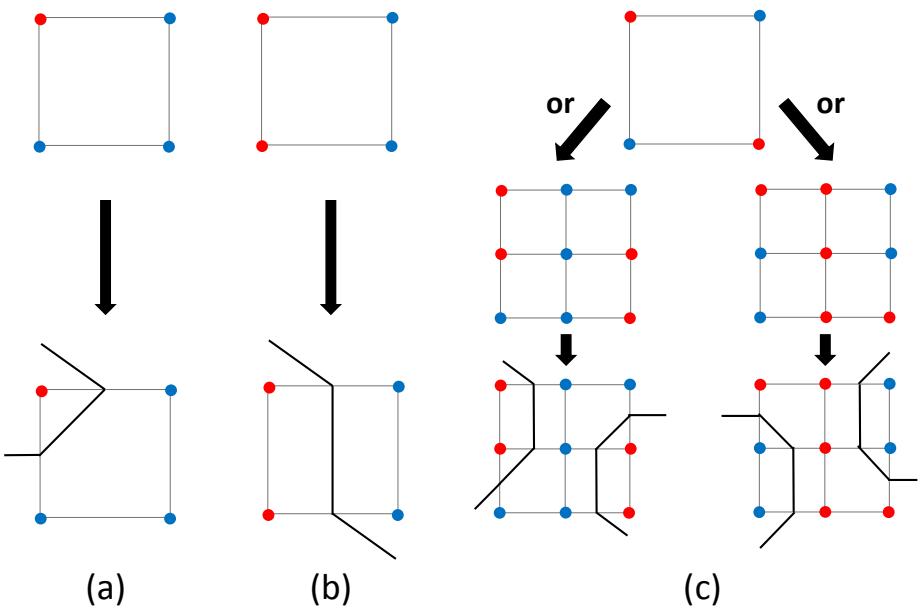

Figure 5.10: Four topological configurations and the corresponding 2-color grid-cells. For example (c), the color of the middle sample after subdivision will typically disambiguate the two cases, unless the subdivision gives rise to another case (c), in which case we continue subdividing those sub-grid-cells.

In configurations (a) and (b), there are two heterogeneous grid-sides. When tracing the Voronoi edges, if the edge being traced enters this grid-cell from one of the heterogeneous grid-sides, it will exit on the other heterogeneous side to the next neighboring grid-cell.

When the sampling density is insufficient, we might have configuration (c). Just as for Marching Cubes [57], there is insufficient information to determine the topology inside this grid-cell. We use the same uniform subdivision as in Section 5.3 step 5 to subdivide this grid-cell into four sub-cells, and get five new colored u-v sample points. If the four sub-grid-cells are all configuration (a) or (b), we can determine the Voronoi edge trajectory inside them. If any of the sub-grid-cell is still in configuration (c), we continue subdividing until no grid-cells have such a configuration, or a maximum depth of recursion is met.

5.5.2 Tracing Voronoi edges via "through-grid-cells"

After preprocessing all the 2-color grid-cells to configuration (a) and (b), all the grid-cells are ready for our edge tracing process. For each base sphere, the search for Voronoi edges is based on the colors of the u-v sample points on its corresponding bounding cube. Our search starts at each grid-cell containing a Voronoi vertex. Such grid-cells are at least 3-color grid-cells, which contain multiple heterogeneous grid-sides with different edge identifiers (color pairs).

Each such color pair indicates a unique Voronoi edge emanating from the Voronoi vertex and exiting to the neighboring grid-cell that shares the grid-side with that edge-identifier (Fig. 5.11(a)). We trace this edge to this next (neighboring) grid-cell. We check if this new grid-cell is a through-grid-cell for the particular edge we are tracing. If so, we identify the other (exiting) grid-side with the same edge identifier as the grid-side through which we entered the grid-cell, and proceed to the corresponding neighboring grid-cell. We keep tracing the Voronoi edge to its next grid-cell, and repeat the process above (Fig. 5.11(b)). In each iteration, we check if the new grid-cell is also a "through-grid-cell" of this edge. If so, we proceed to the neighboring grid-cell sharing the exiting grid-side (with the matching edge identifier), and mark this grid-cell as "traced" for this particular edge identifier color pair. In addition to calculating edge geometry sample points in each iteration, we take the average of the 3D space coordinates corresponding to the two sample points of the exiting grid-side, and using that average location as our start point, run Newton-Raphson iteration (similar to Section 5.3 step 5) to find a point on the Voronoi edge in actual 3D space.

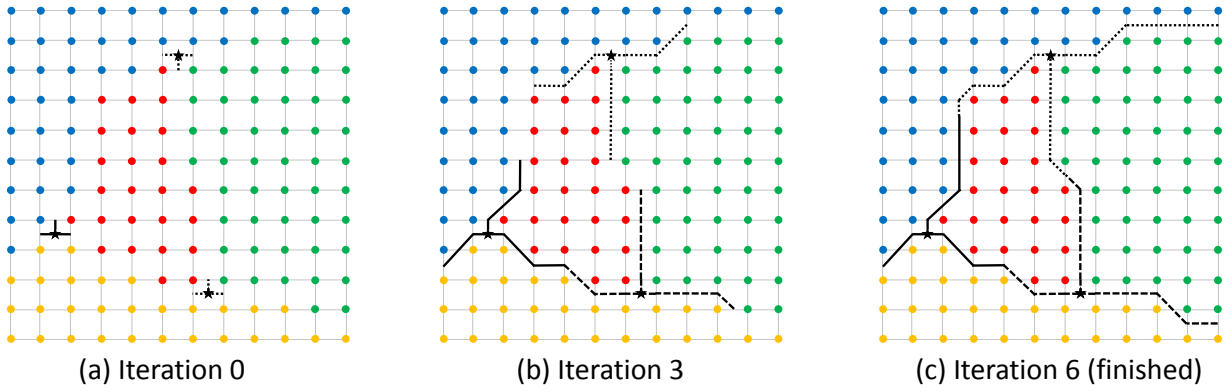

Figure 5.11: Topology construction process on a u-v surface (sampling density 12*12); stars indicate the presence of a Voronoi vertex in the grid-cell. Traces from different vertices are shown with different line styles.

We trace all unique edges from all Voronoi vertices in parallel. Each trace terminates when either of the following conditions is met:

1. The next "through-grid-cell" of the edge is already marked with the same edge identifier, which means it has met up with the search from the other end of the same edge (Fig. 5.11(c)). In this situation, we record the connectivity between the starting Voronoi vertices corresponding to each path, and combine the 3D sample points we calculated along the two paths, reversing the order of points from one trace. Thus we obtain not only the Voronoi edge topology, but also ordered sample point locations on the edge geometry. These points can be used for visualization or analysis.

2. The next grid-cell is not a "through-grid-cell" of the edge, and has one or more sample points at infinity. The existence of sample points at infinity and the absence of an exiting grid-side with the corresponding edge identifier indicates that the Voronoi edge we are tracing goes to infinity. (Fig. 5.3 shows an example with sample points at infinity.) In this situation, we terminate the search, and record the topology information and 3D point locations on the edge geometry of this infinite Voronoi edge. This situation is further discussed in Section 5.5.2.

It is also very rarely the case that the next grid-cell is not a "through-grid-cell" of the edge but does not go to infinity, in which case it needs to be subdivided to continue tracing the edge, as discussed in Section 5.5.2.

Finally, we gather all the information from each base sphere. Each Voronoi edge occurs in the Voronoi cell of at least three base spheres (three for general position, more for non-general position). The 3D point locations on the edge geometry will be different for each base sphere's representation of the same Voronoi edge. Because the choice of 3D points do not affect the accuracy of topology construction, we randomly keep one group of 3D point locations for each Voronoi edge.

Some special conditions may bring more complexity into our tracing process. Although the algorithm we described above is able to handle them, some implementation details should be emphasized. We discuss such conditions below.

Non-uniform grid-cells

Because of the generation of sub-grid-cells from the subdividing operation (e.g. Section 5.3 step 5) and/or from shooting new rays towards matched vertices from base spheres that did not initially find them (Section 5.4.1), sometimes we do not have uniform grid-cells on the parametric bounding cube. Fig. 5.12(a) shows an example of non-uniform grid-cells on a parametric face that had an original sampling density of 2 by 2 cells and had subsequent sub-grid-cells added by both of these operations.

In the edge tracing process, just like with uniform grid-cells, we check if the exiting grid-side (with corresponding edge identifier) exists among all the grid-sides in this non-uniform grid-cells. If it does, we continue tracing to the next corresponding grid-cell, otherwise we subdivide this grid-cell and continue the trace through the new generated sub-grid-cells (details in Section 5.5.2). The updating process in Section 5.4.2 still applies to obtain the neighboring/grid-side information and construct the edge topology on non-uniform grid-cells (Fig. 5.12(b)).

It is necessary to be aware, as described in Section 5.4.2, that on these non-uniform bounding cubes, some grid-cells would have more than 4 neighboring grid-cells. When the initial sampling density is too low, more non-uniform grid-cells will be generated. Grid-cells may even have tens of neighbors. Such poor uniformity would damage the efficiency of our parallel algorithm, so choosing an appropriate initial sampling density is important in the implementation. We will discuss this more in Section 5.6.

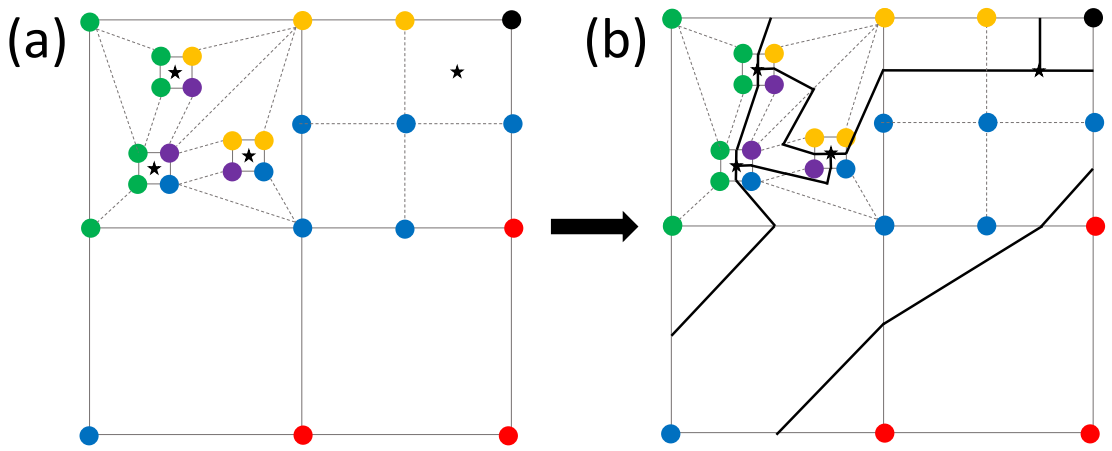

Figure 5.12: Edge topology construction process of a non-general u-v parametric face (a) with an original sampling density 3 by 3. A star indicates the presence of a Voronoi vertex in the grid-cell. (b) The resulting edge trace topology from the Voronoi vertices is shown with bold lines.

Infinite sample points

Voronoi edges do not always terminate at Voronoi vertices. Some Voronoi edges extend to infinity, on one or both ends. We call such Voronoi edges infinite edges.

During our edge tracing process, if the current grid-cell does not have an exiting grid-side with corresponding edge identifier, and has at least one corner sample point at infinity, this Voronoi edge extends to infinity in 3D space. In this situation, we will record that this edge extends to infinity and stop the search.

If a grid-cell contains a sample point at infinity, but still has matched entering and exiting grid-sides, we should continue tracing the Voronoi edge through the grid-cell neighboring at the exiting grid-side (Fig. 5.13).

Additional subdivision for topological disambiguation

During the edge tracing process, the trace might enter a grid-cell that does not have a clearly matched exiting grid-side, yet has no sample points at infinity. This situation happens when the sampling density is insufficient. There are two cases: the grid-cell has (1) no other grid-sides with the corresponding edge identifier; or (2) multiple grid-sides with the corresponding edge identifier.

An example of the first case is shown in Fig. 5.14(a), where the target Voronoi edge is close to another edge but not intersecting with it. In this example, if we are tracing the upper Voronoi edge from the right to the left, the edge enters grid-cell 1 from its right grid-side e. Since there are no other grid-sides with the red-blue edge identifier, grid-cell 1 is

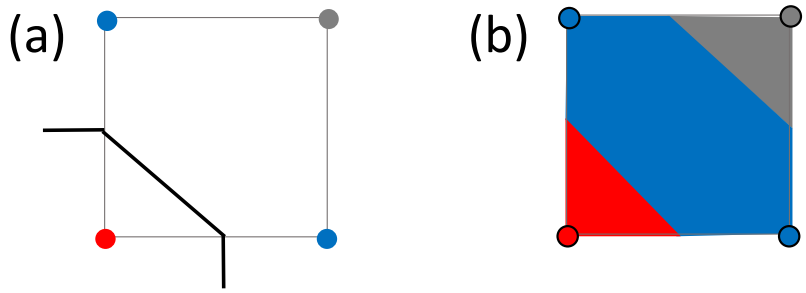

Figure 5.13: (a) The tracing path in a grid-cell containing an infinite sample point at infinity and a new grid-side to update. (b) The actual color pattern inside the grid-cell (grey represents infinity).

not a "through-grid-cell"; we need a greater local sampling density in grid-cell 1 to continue tracking this Voronoi edge.

In this situation, we subdivide the current grid-cell, then continue tracing on the appropriate new sub-grid-cells by checking the color identifier over the two new sub-grid-sides generated from the previous entering grid-side. In Fig. 5.14(b), we subdivide grid-cell 1, then continue tracing on sub-grid-cell (1) at a new entering grid-side e_1. The same situation occurs when the trace enters grid-cell 2 and grid-cell 3; we repeat the subdivision process, iteratively subdividing the two grid-cells when the trace enters each of them, and get the edge path as shown in Fig. 5.14(c). If necessary, we repeat the subdivision process until all the grid-cells along this trace are "through-grid-cells," or a maximum depth of subdivision is reached.

At the maximum recursion depth, if we still cannot distinguish their respective paths, we treat the edges as coincident in this neighborhood. In Fig. 5.14(d), assume all the grid-cells are already at max recursion depth. When tracing to grid-cell 1, no other grid-sides match either of the blue-red or blue-yellow edge identifiers corresponding to the two entering traces. Under the local sampling density at the maximum recursion depth, the two edges are still too close to each other, with no sample point of the color (blue) detected on the exiting grid-side (the left grid-side of grid-cell 1). In this case, we treat the two edges as coincident in the neighborhood of grid-cell 1, and trace the exiting yellow-red-(blue) "super edge." We keep tracing this super-edge by the yellow-red edge identifier, until the edge identifiers of both the two individual edges (blue-red and blue-yellow) re-occur (in grid-cell 4). We then continue each of the individual traces of those two edges.

An example of the second case that requires disambiguation, where the grid-cell has multiple possible exiting grid-sides with the same edge identifier, is shown in Fig. 5.15. We subdivide such grid-cells with the same process as the first case, unless it is a 2-color grid-cell in configuration (c) (Fig. 5.10(c)), and already met the maximum depth of recursion (Section 5.5.1). To disambiguate the true layout in that case, we will compare the edge

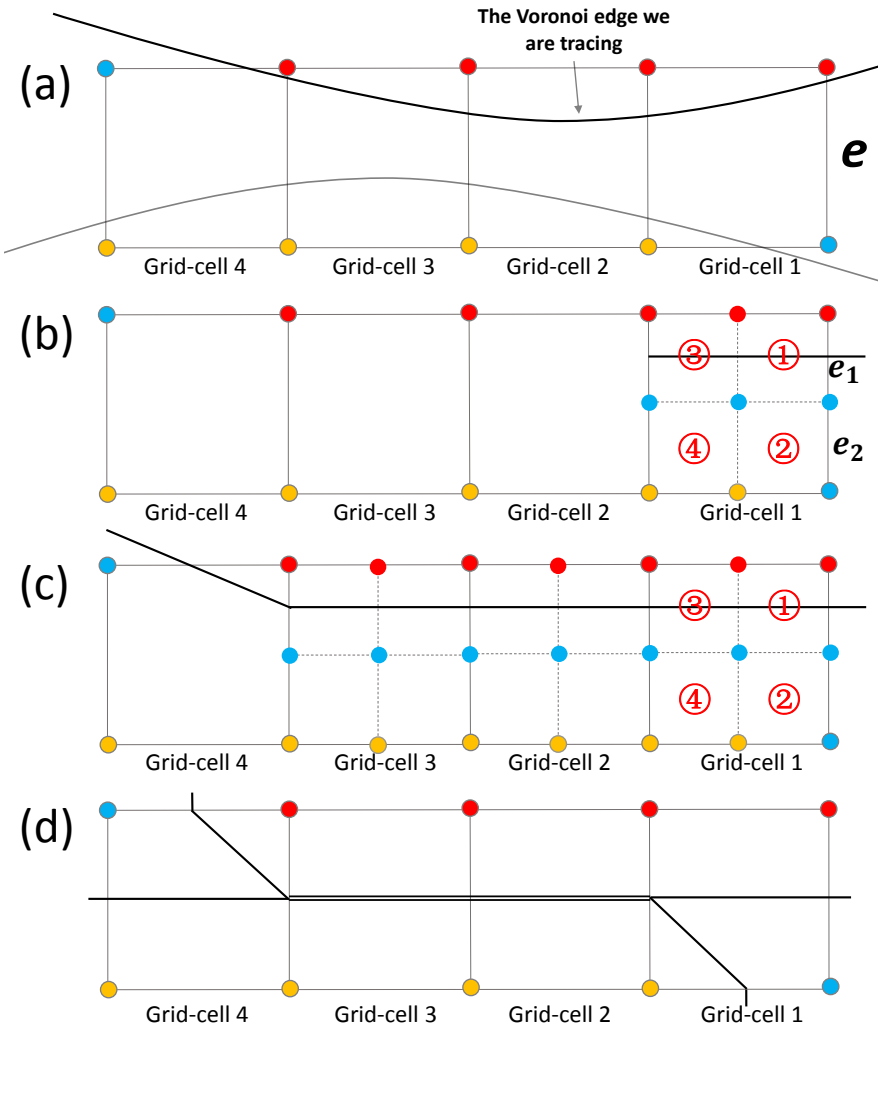

Figure 5.14: The subdivision operation on grid-cells that are neither "through-grid-cells" nor contain sample points at infinity.

topology result from two possible layouts to the results from the other Voronoi cells that share the edge (Section 5.5.4).

Case of u-v deviation

Under insufficient sampling density, there might be two or more Voronoi edges entering a grid-cell through different grid-sides and exiting through the same grid-side, without intersecting with each other inside this grid-cell. In such situations, the grid-cell might be 3-color but has no Voronoi vertices located in its corresponding 3D space neighborhood. An example is

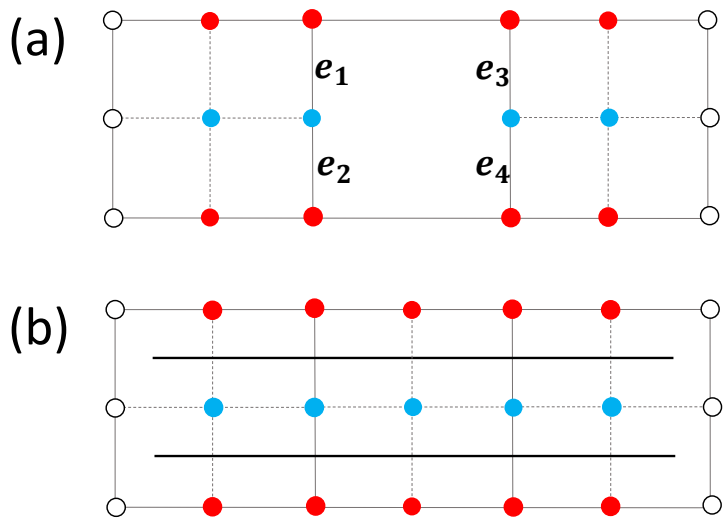

Figure 5.15: An example of a grid-cell that has more than two grid-sides with the same red-blue identifier: (a) when the edge tracing enters the middle grid-cell by any of the grid-side e_1, e_2, e_3, or e_4, it will have three other grid-sides with the same (red-blue) edge identifier; (b) the result of the edge tracing process after the middle grid-cell is subdivided.

shown in Fig. 5.16(a), where two Voronoi edges enter grid-cell 2 through different grid-sides (the top and the right grid-sides) and both exit through the same grid-side (the left grid-side). It is a 3-color grid-cell, but the corresponding vertex geometry is not within its u-v sub-domain. If we were to shoot a ray directly to this Voronoi vertex, it would be in the u-v sub-domain of grid-cell 1. However, because of missing the red color, grid-cell 1 only has two colors, indicating no existence of Voronoi vertices in the corresponding 3D space.

We call this situation a "u-v deviation," because instead of being located in the u-v sub-domain of a ray to the actual Voronoi vertex geometry (grid-cell 1), the vertex location "deviates" to a nearby 3-color grid-cell with the correct color codes corresponding to this Voronoi vertex.

Although we may have such "u-v deviations," the 3D space geometry and topology information for such Voronoi vertices are still obtained correctly. Our 3D space geometry calculation is based on the bisector equations among the generating spheres (Section 5.3 step 5). With the same base sphere and 3-color pattern, starting from a different grid-cell around the true u-v location only means an offset of the iteration start point; the numerical iteration still finds the correct 3D geometry of the vertex. For the topology, as shown in Fig. 5.16(b), the Voronoi edges correctly connect to the corresponding 3-color grid-cell, even though the grid-cell containing the actual u-v location of the ray to the vertex is the neighboring grid-cell (1).

In degenerate cases, if a Voronoi vertex is shared by more than three Voronoi edges,

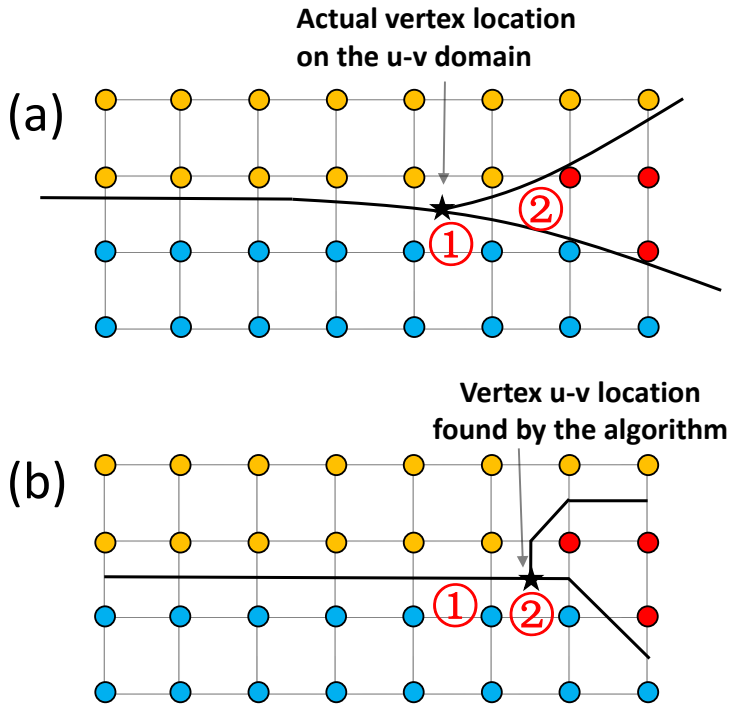

Figure 5.16: (a) The actual vertex and edge location in the u-v domain. (b) The calculated vertex and edge location in the u-v domain by our algorithm.

multiple corresponding 3-color grid-cells for the vertex locations will typically exist. In Fig. 5.17(a), a Voronoi vertex is shared by six Voronoi edges. In the u-v domain, there exists four 3-color grid-cells containing vertices (Fig. 5.17(b)). In the Voronoi vertex sorting process (Section 5.3 step 6), we will determine that these four vertex locations correspond to the same Voronoi vertex because all the calculated vertices have the same 3D space coordinates.

In our edge tracing process, connecting to any of these four 3-color grid-cells is treated as connecting to this particular Voronoi vertex. The zero-length "Voronoi edges" between Voronoi vertices in 3-color grid-cells corresponding to the same actual Voronoi vertex, allow them to be merged when calculating edge topology. As seen in Fig. 5.17, "u-v deviation" is what allows us to handle such high-order Voronoi vertices. Under non-general position input, a Voronoi vertex may deviate to multiple corresponding grid-cells on the parametric bounding box, extending the ability of our algorithm to detect the vertex's connectivity with more than four Voronoi edges (even though a non-subdivided grid-cell can at most detect four Voronoi edges through its four grid-sides).

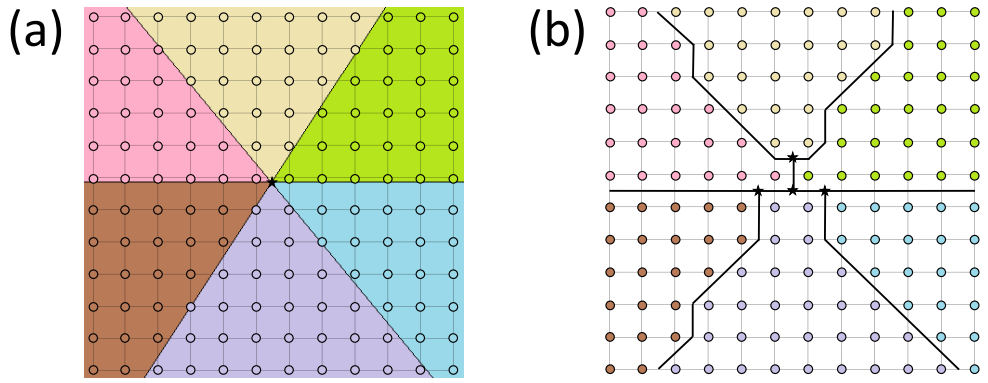

Figure 5.17: A high-order Voronoi vertex shared by six Voronoi edges: (a) the actual vertex and edge location on the u-v domain; (b) the calculated vertex and edge location on the u-v domain by our algorithm.

5.5.3 Detecting isolated Voronoi edges

After the edge tracing process described in Section 5.5.2, we will have constructed the topology of all the Voronoi edges that connect Voronoi vertices.

However, not all Voronoi edges are connected with Voronoi vertices. Two types of isolated Voronoi edges are disconnected from any of the Voronoi vertices, and we cannot find them from our general edge tracing process. They are:

1. Infinite Voronoi edges with both ends extending to infinity. An example of this situation is shown in Fig. 5.18, in which five spheres have their center on the same plane. If we look at the white sphere's parametric bounding cube (Fig. 5.18(c)), there are four Voronoi edges with both of their ends corresponding to grid-cells with infinite sample points.

2. Self-connected Voronoi edges. As shown in Fig. 5.1(a), this ring-like Voronoi edge does not have any actual endpoints.

In (Section 5.5.2), starting from each of the Voronoi vertices, we marked all the "through-grid-cells" with corresponding edge identifiers along each tracing path. After the tracing process, we check if each "through-grid-cell" has been marked for each distinct edge identifier of all its grid-sides. If a "through-grid-cell" is unmarked for any of its edge identifiers, this "through-grid-cell" is related to an isolated Voronoi edge corresponding to the unmarked edge identifier.

We collect all such unmarked through-grid-cells. First we sort them into different groups by their unmarked edge identifiers. If an unmarked "through-grid-cell" has multiple edge identifiers, each edge identifier represents a particular isolated edge; it will be put in all the corresponding groups. For each group, we randomly choose one of its grid-cells as our starting

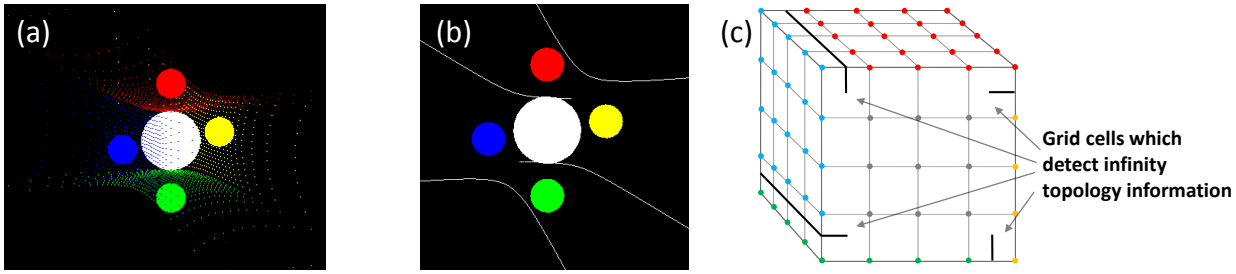

Figure 5.18: An example of inputs generating infinite isolated Voronoi edges (grey represents infinity).

grid-cell, and start tracing the paths of the edge through the two grid-sides corresponding to the two edge identifier colors of this through-grid-cell. In each trace, we repeat the same iteration process as for edge tracing (Section 5.4.1). For the search for each isolated edge, if the two traces both stop at grid-cells indicating infinity (Section 5.5.2), the Voronoi edge is an infinite Voronoi edge with both ends extending to infinity. If the two traces meet each other, the Voronoi edge is a self-connected edge.

5.5.4 Sorting of the Voronoi edges

After calculating all the Voronoi edges (including Voronoi edges connected with vertices or isolated Voronoi edges) from each Voronoi cell (base sphere), we combine the edge information of each individual Voronoi cell to form the whole Voronoi diagram by sorting and merging the Voronoi edges detected for each base sphere's Voronoi cell.

A Voronoi edge has at least three contributing spheres (three for general position and four or more for non-general position), so it will have at least three "edge uses" in the Voronoi cells for those spheres. Each Voronoi edge use has its own 2-color edge identifier, and one color code corresponding to the base sphere of its Voronoi cell. We call this unique Voronoi cell related color code the "cell identifier" of this Voronoi edge use. The color triplet, indicates the three contributing input spheres of the corresponding Voronoi edge.

We sort all the Voronoi edge uses by these corresponding color triplets and the Voronoi vertices they connect (typically two, but special cases of self-connected Voronoi edges with zero or one Voronoi vertices may exist). For each sorting group with the same color triplet and corresponding Voronoi vertices, if there are three Voronoi edge uses in the group and each of them has a different cell identifier, the particular Voronoi edge corresponding to this group exists between the corresponding Voronoi vertices (or infinities), and can be found in all of its corresponding Voronoi cells. A rare case occurs when two Voronoi edges exist between the same corresponding pair of Voronoi vertices, and both of them were found in the same corresponding Voronoi cells (Fig. 5.19). In this case, six Voronoi edge uses are in one sorting group, and correspond to two distinct Voronoi edges.

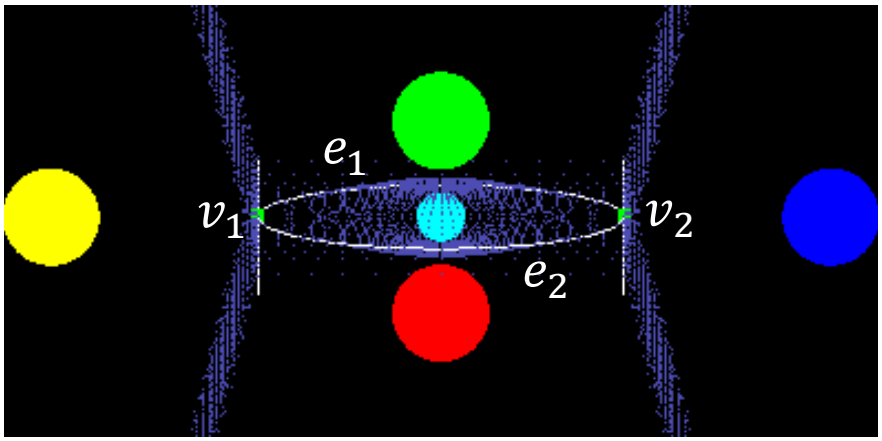

Figure 5.19: An example of two Voronoi edges (e_1 and e_2) sharing the same three contributing spheres (the green, cyan, and red spheres) and two Voronoi vertices (v_1 and v_2) they connect.

By this sorting process, we are able to find all the general-position Voronoi edges, self-connected Voronoi edges, and infinite Voronoi edges in the Voronoi diagram. However, "high-order" Voronoi edges shared by more than three cells will remain unmatched, requiring a second round of sorting.

For any of the Voronoi edge uses not satisfying the conditions in the first sort, we pick one Voronoi edge use and its color triplet as the first member in its group. Starting from this triplet, we iteratively search the remaining unmatched Voronoi edge uses for those where two of the three colors in the triplet match any member triplet colors in the group and have the same corresponding Voronoi vertices to all the members, adding the corresponding Voronoi edge use as a new member in this group if so. After the search for the first group, we repeat this process for any remaining un-grouped Voronoi edge uses, until all of them are grouped. In each group, if the number of different colors equals to the number of Voronoi edge uses, and each of them has a different cell identifier, the particular high-order Voronoi edge corresponding to this group exists between the corresponding Voronoi vertices (or infinities), and can be found in all of its corresponding Voronoi cells.

After this second round of sorting, the only unmatched Voronoi edge uses should correspond to ambiguous grid-cells at maximum subdivision depth (Section 5.5.2). Such grid-cells had two possibilities, only one of them is a true Voronoi edge. For such a pair of Voronoi edge uses corresponding to the same ambiguous grid-cell, if only one of them matches with unmatched Voronoi edge uses from other Voronoi cells and can be grouped with them in a consistent Voronoi edge, we are done and stop considering the other possibility. If neither can be grouped consistently, additional subdivision is needed to disambiguate the corresponding grid-cell.

After these two rounds of sorting, all the Voronoi edge uses should be grouped into corresponding Voronoi edges in the Voronoi diagram; otherwise we restart the algorithm,

increasing the initial sampling density and the maximum recursion depth. Similarly, during the edge-tracing process, if the maximum depth of recursion is met in any step (Section 5.4.1, 5.5.1, or 5.5.2), we restart the algorithm with a higher initial sampling density and maximum recursion depth.

5.6 GPU framework

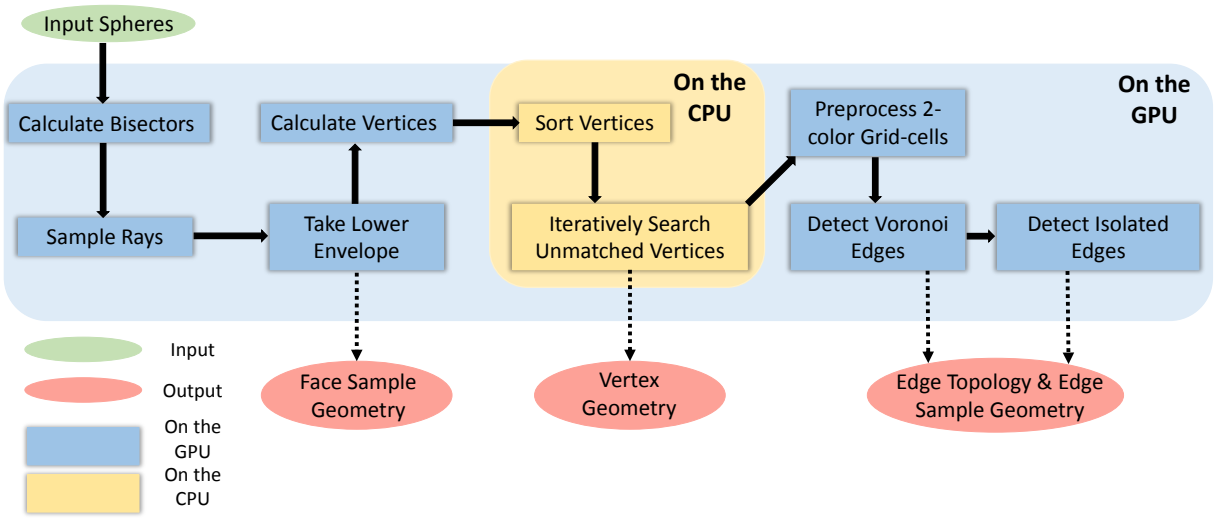

Figure 5.20: The GPU Framework.

Most steps of our algorithm are implemented on the GPU (Fig. 5.20) using CUDA programming to exploit data parallelism. For example, in the edge detecting step, for all paths we trace (starting from all the vertices on all u-v bounding cubes), we use the same operation that iteratively finds neighboring grid-cells sharing the same heterogeneous grid-side. In CUDA, each unit of data is processed on one GPU thread. The method/function being executed by all the GPU threads in parallel is called a kernel. Table 5.1 summarizes the kernel and thread information of each step performed on the GPU.

5.7 Results

Our algorithm to compute the whole Voronoi diagram (geometry and topology) was run on a PC with an Intel® Core™ Processor i7-9700K CPU with 16GB RAM and an NVIDIA GeForce GTX 1080 Ti graphics card. To test our ability to handle large real-world inputs, we implemented our algorithm on protein structures from the protein data bank [58], where protein molecule structures are represented as combinations of atom spheres with different radii.

Table 5.1: Thread and kernel information for all steps performed on the GPU. Colors correspond to the timing breakdown of geometry/topology in Fig. 5.21.

Step	Per Thread	Kernel
Calculate Bisectors	Each input sphere	Calculates all the bisectors between each sphere and all other input spheres (Section 5.3 Step 1)
Sample Rays	Each ray	Samples the rays from all input spheres (Section 5.3 Step 2)
Take Lower Envelopes	Each ray	Calculates the intersections between each ray and all bisectors, keeping the intersection point with minimum ray distance (Section 5.3 Step 3)
Calculate Vertices	Each grid-cell	Finds the grid-cells containing Voronoi vertices and calculate the vertices by numerical iteration (Section 5.3 Step 4 and 5)
Preprocess 2-color Grid-cells	Each grid-cell	Checks if the grid-cell is 2-color and in configuration (c), and subdivides such grid-cells to configuration (a) and (b) (Section 5.5.1 Fig. 5.10)
Detect Edges	Each trace	Starting from each grid-cell containing a Voronoi vertex, for each edge identifier in such grid-cells, the kernel traces the corresponding Voronoi edge via "through-grid-cells" (Section 5.5.2)
Detect Isolated Edges	Each trace	Starting from a random member of each group of unmarked "through-grid-cells" with the same edge identifier, the kernel traces the corresponding isolated Voronoi edge (Sections 5.5.2 & 5.5.3)

In the implementation of our algorithm, the selection of appropriate sampling density (the number of u-v samples on each face of bounding cubes) is important for obtaining good parallelism. If the sampling density is too low, we will not obtain enough information to find Voronoi vertices, and then in the calculating vertices step, we will need to subdivide many grid-cells and reconstruct their neighboring information. Subdivisions will damage the uniformity of the grid-cells, reducing data parallelism, and decreasing the efficiency of our algorithm. On the other hand, if the sampling density is too high, we obtain too many sample points that are unnecessary for finding Voronoi vertices and edges (such as sample points on 1-color grid-cells). To illustrate this trade off, we ran our algorithm on protein "1crn-PDB" with 327 input spheres and different sampling densities. Table 5.2 shows how the total number of grid-cells increases along with the increase of sampling density, but the number of subdivision operations decreases. Total number of grid-cells includes original grid-cells and sub-grid-cells generated by subdivision and targeted search. Fig. 5.21 shows how the run time varies with sampling density for the same input. Among all the proteins tested, sampling density in the range 40*40 to 50*50 provides the lowest total run time. When the sampling density is lower than 40*40, the steps up to and including calculating vertices (steps of geometry calculation on GPU) take an extremely long time because of the lack of parallelism in the subdivision operation. When the sampling density is higher than 50*50, the parallelism of our algorithm is excellent but the increasing number of unnecessary sample points hurts the run time.

Table 5.2: Number of subdivisions and deepest level of subdivision with different sampling densities, for protein 1crn-PDB with 327 input atoms. Total number of grid-cells includes original grid-cells and sub-grid-cells generated by subdivision and targeted search.

Sampling Density	**# of Subdivision Operations**	**Total # of Grid-cells**	**Deepest Level of Subdivision**
1*1	4,237	15,046	8^{th}
10*10	405	197,945	5^{th}
20*20	195	785,660	4^{th}
40*40	81	3,139,584	3^{rd}
80*80	23	12,556,922	2^{nd}
160*160	3	50,227,212	1^{st}
320*320	0	200,908,800	N/A

On other protein models we tested containing 217 to 4195 spheres, the run times also indicated that sampling densities from 40*40 to 50*50 were the most efficient (lowest overall run time).

The total computation time of our algorithm under different input sizes is shown in Fig. 5.22. When the sampling density is lower (e.g. 10*10 here), the computational efficiency is inherently dependent on the geometric distribution of the input spheres, which determines the data parallelism (number of subdivision operations) of our algorithm. When the sampling

density obviates most subdivision (usually more than 40*40), the computation time increases roughly linearly with the number of input atoms (spheres).

To test our ability to handle non-general position inputs, we designed example inputs in three non-general situations: self-connected Voronoi edges, infinite Voronoi edges with both ends extending to infinity, and high order Voronoi vertices or edges. Our algorithm successfully handled all the non-general situations; one result from each of situation is shown in Fig. 5.1.

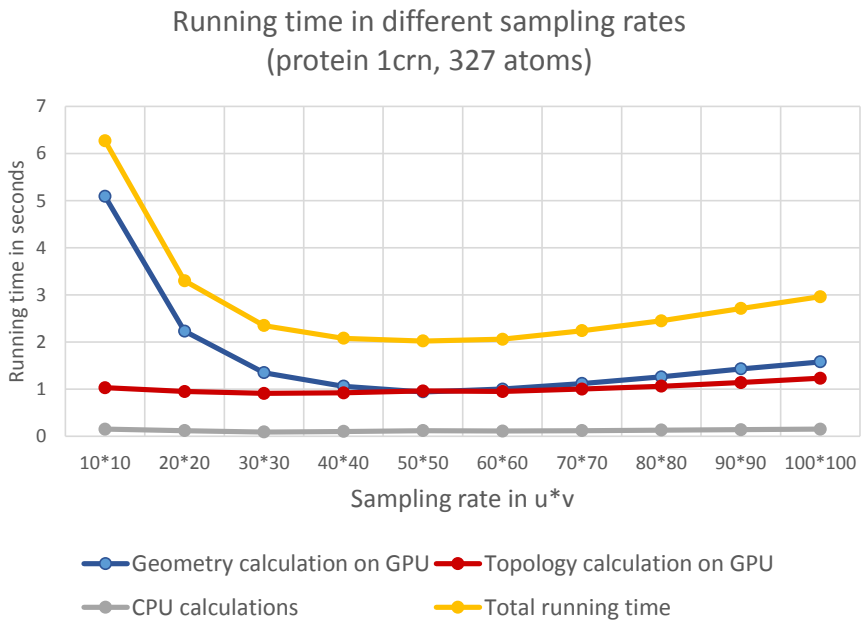

Figure 5.21: Run time vs. sampling rate, protein 1crn-PDB with 327 atoms.

For the verification of our experimental results, we implemented a naive brute-force algorithm for detecting all Voronoi vertices, to serve as ground truth. The algorithm is based on the fact that a Voronoi vertex always corresponds to a sphere that is tangent to all of its (four or more) contributing spheres, and does not intersect or contain any other input sphere.

Therefore, for each combination of four spheres from the input, we calculate their tangent sphere using the algorithm described by Gavrilova and Rokne [59], and check if the resulting tangent sphere intersects (tangent not included) or contains any of the other input spheres. If not, the center of this tangent sphere is a Voronoi vertex. We exhaustively make this check for all the four-sphere combinations and gather the information of all the calculated Voronoi vertices.

In our experiments, all of the results produced by our algorithm matched the results from this brute-force algorithm. Furthermore, in all cases, including large-size inputs from the

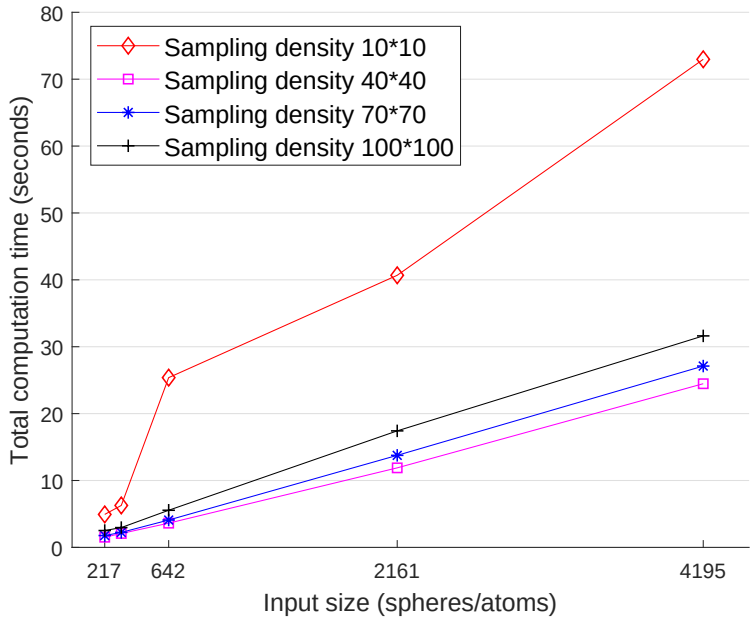

Figure 5.22: Computation time at different sampling densities on protein models: (1) "1al1-PQR" with 217 atoms; (2)"1crn-PDB" with 327 atoms; (3)"1crn-PQR" with 642 atoms; (4)"1bh8-PQR" with 2161 atoms; and (5)"1JD0-PDB" with 4195 atoms.

protein data bank (the five shown in Fig. 5.22 and ten other randomly selected proteins) and the non-general position cases, the Voronoi edge calculation is consistent among all the Voronoi cells (meaning there are no unmatched Voronoi edge uses after two rounds of sorting). Futhermore, the same output Voronoi diagram vertices and edge topology is produced for all sample densities that we tested, even with the coarsest possible initial 1x1 sampling (just the 8 corners of each bounding cube). In the tests, we set the maximum subdivision depth to 10, and this bound was never met (Table 5.2); in other words, less than 10 levels of subdivision was fine enough to trace all the Voronoi edges.

We apply the 2D version of this algorithm in our resin-rich area detection. After substituting this method for the previous exact computation (cell expansion) method, for the same real-world input mentioned in Section 5.1, the computation time of the Voronoi construction process decreases from 61.33s to 22.69s (the overall resin-rich area detection time decreases from 74.48s to 37.07s). Both methods accurately calculate Voronoi diagrams and resin-rich areas.

Another potential sample-based approach to constructing Voronoi diagrams would be to approximate input circles or ellipses by sampling points on their boundaries [60], constructing a point-set Voronoi diagram, and merging Voronoi cells corresponding to the same input circles or ellipses. This method avoids complex exact calculations, but may lead to accuracy

problems and bring additional complexity to the automatic detection process (e.g. in choosing the sampling / approximation density).

5.8 Conclusion

We have presented a sample-based algorithm to construct edge topology for Voronoi diagrams of spheres in $\mathbb{R}^3$. It successfully handles input spheres in both general and non-general position, including self-connected Voronoi edges, infinite Voronoi edges, and high-order position inputs. We design a GPU framework to exploit data parallelism, and find the approximate range of sampling densities to maximize the efficiency of our algorithm.

By integrating the 2D version of this method in our resin-rich area detection, we obtain the same (accurate) detection result as the exact computation method, and reduce the total computation time by over 50%. The result verifies that in the resin-rich area detection process, sample-base geometric processing algorithms are able to achieve the same level of accuracy as exact computation methods, but with a much shorter computation time.

Chapter 6

Defect Detection from Microscope Images: Distance Transform Approach

6.1 Introduction

In Chapter 5, by applying a sample-based approach in the Voronoi construction step, we reduce the computation time of resin-rich area detection by over 50%. However, the computation of Voronoi diagrams is still the most time-consuming step, taking about 60% of the implementation time.

In this chapter, we propose a novel sample-based algorithm to automatically identify resin-rich areas from composite cross-sectional images by utilizing the concept of the α-hull, bypassing the calculation of Voronoi diagrams. As discussed in Section 2.2, the α-hull is the close relative of the α-shape, and its complement can be used to characterize the resin-rich areas (Fig. 6.1). Beyond the Voronoi approach, our new method is not only able to handle input images containing arbitrary fiber cross-section shapes, but also significantly increases the computation efficiency, with our design of a new α-hull construction procedure without utilizing Voronoi diagrams.

Our main contributions include:

- A mathematical approach to precisely define the boundaries of resin-rich areas based on the concept of α-hulls.
- A robust and efficient method to calculate the α-hull from inputs of arbitrary shapes, not just point set inputs, via the distance transform and morphological dilation operation.
- A novel end-to-end algorithm to automatically identify resin-rich areas from composite cross-sectional images that is:

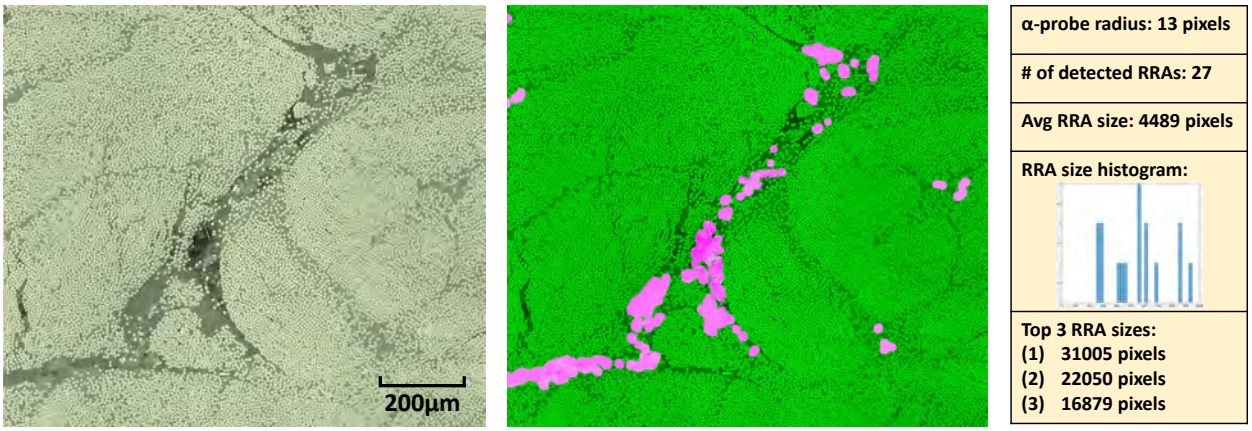

Figure 6.1: An example experimental result of our algorithm: (a) input cross-sectional image (1423*1623 pixels, a small subsection of the full microscope image; subsection contains about 15 thousand fiber cross-sections); (b) algorithm output image, resin-rich areas detected (shown in semi-transparent pink); (c) statistics of detected resin-rich areas (RRAs), please refer to Section 6.3.2 for more details.

- flexible: handles fiber cross-sections of arbitrary shapes and sizes;
- fast: for high-resolution real-world microscopic images (approximately 18,000*10,000 pixels) with about 1.15 million fiber cross-sections, calculates resin-rich areas in 3.5 seconds. Compared to the Voronoi approach results reported in Chapter 4, our new proposed approach reduces the computation time by 95.3%.

Voids (or porosities) are also a major defect in FRP composites. Voids that exist within or attached to resin-rich areas may have originally been resin-rich areas, for example, air entrapment in the resin-rich areas generated while abrading and/or polishing the microscope sample surfaces [21]; and regardless, voids are more similar to resin-rich areas than fibers because both are un-reinforced areas. In this chapter, we define resin-rich areas to include any voids, but our algorithm implementation is also able to identify and distinguish void regions if desired.

6.2 Calculation of resin-rich areas

In this section, we describe how our algorithm automatically detects resin-rich areas from composite cross-sectional images (Fig. 6.2). First, we convert the original input to a binary image that distinguishes fiber and matrix regions (Section 6.2.1). We treat the fiber pixels as background pixels to build the distance transform of the binary image (Section 6.2.2).

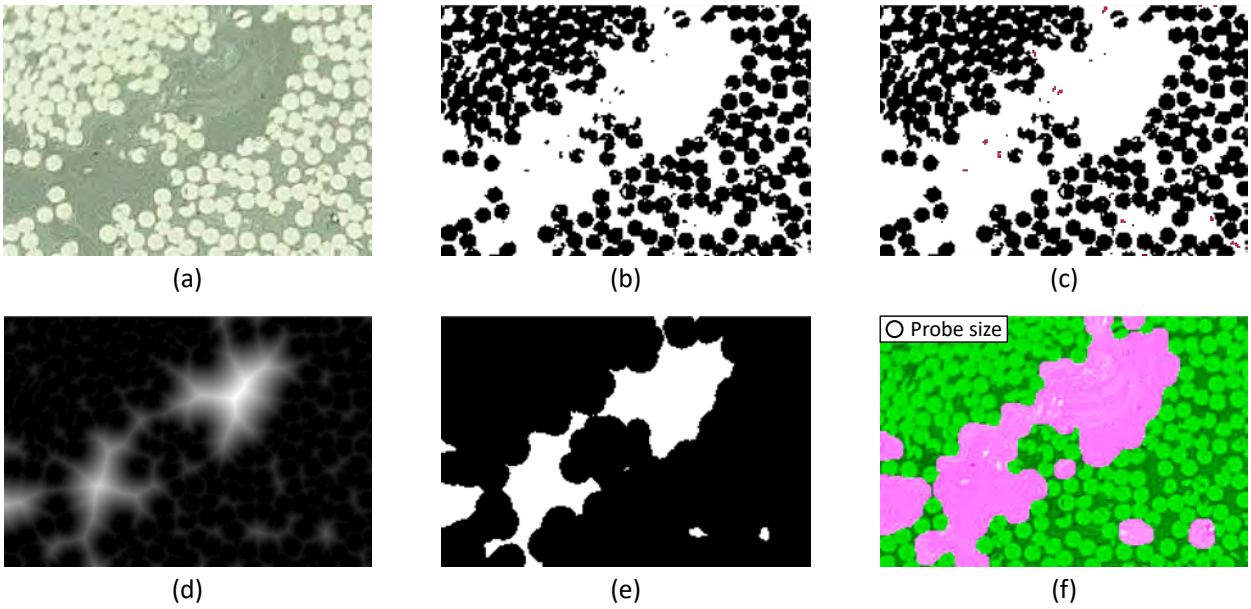

Figure 6.2: Algorithm overview: (a) input microscope image; (b) binarized image; (c) clean image, noisy pixels are identified (in red) and removed; (d) distance transform applied to image; (e) free space for the center of the α-probe (in white); (f) detected resin-rich areas (in semi-transparent pink).

Finally, we construct the complementary areas of the α-hull from the distance transform, which are our identified resin-rich areas. (Section 6.2.3).

6.2.1 Image binarization

In the input cross-sectional images (Fig. 6.2(a)), we distinguish fiber and matrix regions by their pixel intensities. Since the fiber pixels usually have much higher pixel intensities than the matrix pixels, we segment/binarize the image into fiber regions and matrix regions (Fig. 6.2(b)) using image thresholding techniques.

In our implementation, we use Otsu's method [41] to segment the input images. Otsu's method is a global thresholding method that determines the optimal threshold value by maximizing inter-class variance. It performs well on high quality images. Since the input cross-sectional images in our experiments are high resolution microscope images with little noise, Otsu's method is suitable for our analysis.

However, if the input images have relatively low quality, such as cross-sectional images from non-destructive methods (for example, computed tomography), adaptive thresholding [61] or iterated conditional modes [62] that are designed for thresholding noisy images are likely to provide better segmentation results than Otsu's method. If the input images have

insufficient contrast between the fiber and matrix regions, contrast enhancement techniques such as [63, 64] can be applied prior to this image binarization step to maintain the effectiveness of the thresholding methods.

6.2.2 Distance transform calculation

After binarization, the image is denoised (Fig. 6.2(c)) because the distance transform is sensitive to noise. There might be two forms of noise in the image: (1) isolated random image noise and (2) noise representing impurities such as broken fiber debris and unexpected inclusions in the composite. Broken fiber debris and unexpected inclusions are treated as noise because they are not a substantial source of material strength compared to complete fibers in continuous FRP composites. Noise pixels are eliminated by removing (fiber or matrix) regions that have less than a threshold number of connected pixels. In this implementation, the threshold is set as 15% of the nominal size of fiber cross-sections because the impurity sizes are relatively small in the test images. This threshold should be tuned according to impurity sizes in input images.

The nominal size of fiber cross-sections can be determined (or estimated) using one of the following methods: (1) converting the real nominal size (if available) to pixel units using the resolution of the microscope; (2) approximating the nominal size by calculating the mode size of the (circular) fiber cross-sections via circle detection methods such as [45]; or (3) estimating the nominal size of the fiber cross-sections by manual observation. If the fiber cross-sections are not circular (for example, kidney shapes), the nominal size is estimated by calculating the mode size of their minimum enclosing circles.

After removing the noisy pixels, we build the distance transform of the resulting binary image using the algorithm described in [42] by setting matrix pixels as foreground and fiber pixels as background (Fig. 6.2(d)).

6.2.3 Construction of α-hull complement

Our mathematical definition of resin-rich areas is the complementary areas of the α-hull, or in other words, those areas that can be reached by the α-probe. Since the α-probe is a circle with a fixed radius α, if the α-probe does not collide with any of the fibers (fiber pixels), the distance between the probe center pixel and its closest fiber pixel must be larger than the radius α.

In the distance transform, the distance value of each pixel is its distance to the closest fiber pixel. Pixels with distance values greater than the α value are pixels where the α-probe center can be located without colliding with the fibers. We call the union of such pixels the free space for the α-probe centers (Fig. 6.2(e)).

Finally, we implement the morphological dilation operation by setting the free space for the α-probe centers as the input shape A, and the α-probe as the structuring element B. $A \oplus B$ gives us the union of all the areas covered by the α-probe when its center moves inside

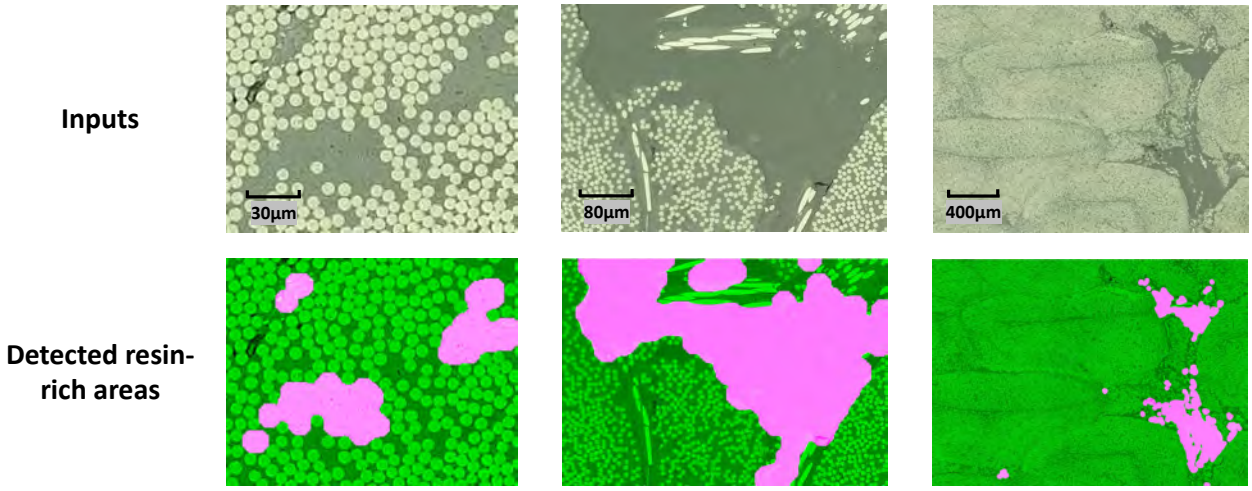

Figure 6.3: Experimental results of our algorithm. Detected resin-rich areas are shown in semi-transparent pink.

the free space, thus matching our definition for resin-rich areas. Therefore the regions our algorithm outputs are the resin-rich areas (Fig. 6.2(f)).

6.3 Experimental results and discussion

As illustrated in Fig. 6.1, 6.3, 6.5, and 6.6, our proposed algorithm robustly detects resin-rich areas from test images. In this section, the following experimental details and algorithm implementation are discussed: the source of the data, resin-rich area statistics, α value choice, algorithm efficiency, and void defect detection.

6.3.1 Data

In our experiments, 30 microscope cross-sectional images from FRP 3D printed parts are used in order to test the proposed algorithm. The 3D-printed composite parts consist of Polyether ether ketone (PEEK) matrix and unidirectional continuous carbon fibers with a nominal diameter of 7 μm, manufactured by Arevo Inc.'s Aqua composite 3D printer [65] using the direct energy deposition technique. Transverse cross-sectional samples are taken from representative regions of the 3D-printed parts, and are polished by a Buehler Ecomet 250 polishing wheel with SiC grinding papers, achieving a surface roughness of 0.05 μm. Test images are created using a high-resolution Keyence VHX6000 digital microscope (0.7 μm per pixel) on the polished cross-sectional samples. We report the size of the test images and detected resin-rich areas in pixels, which can be easily converted to the physical lengths by considering the resolution of the microscope. The test images are roughly 18,000

* 10,000 pixels and have about 1.15 million fiber cross-sections (see example in Fig. 6.4). In order to clearly show the fiber cross-sections and the resin-rich areas, most figures have been cropped to only show small representative portions of the original images. In the test images, the majority of the fiber cross-sections are circular and close to the nominal size, but there are also a fair number of ellipses (misaligned fibers) and other arbitrary shapes (deformed and broken fibers).

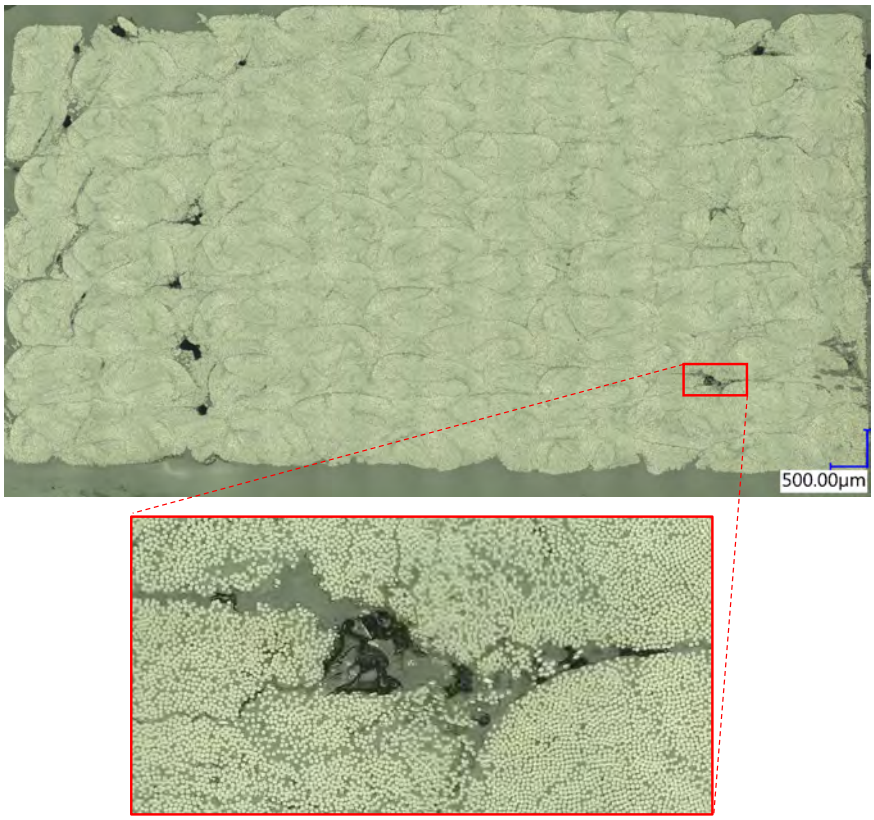

Figure 6.4: An example test image (top image: 18,270*10,306 pixels), the details of the fiber cross-section distributions and resin-rich areas can be better observed by zooming in on local regions (bottom image).

6.3.2 Resin-rich area statistics

The mathematical definition and calculation of resin-rich areas enables us to further distinguish each of the discrete resin-rich areas and summarize their statistics. The detected resin-rich areas from our algorithm (Section 6.2) are represented as a collection of pixels. As shown in Fig. 6.5(a), each contiguous resin-rich region with connected pixels is identified as a separate resin-rich area, which does not overlap or connect with other resin-rich areas.

MATLAB's Image Processing Toolbox is used to label these different continuous regions from calculated resin-rich area pixels, and provide their corresponding areas. Using this information, resin-rich area statistics are calculated, such as the number of resin-rich areas, average resin-rich area size, and a histogram of resin-rich area sizes (Fig. 6.1(c)). This method is able to sort the resin-rich areas by size and visualize the largest ones (Fig. 6.5(b)) as a reference for users. These statistics and visualizations facilitate better understanding of the resin-rich areas, and open up the possibility to further explore their quantitative relationships to material properties.

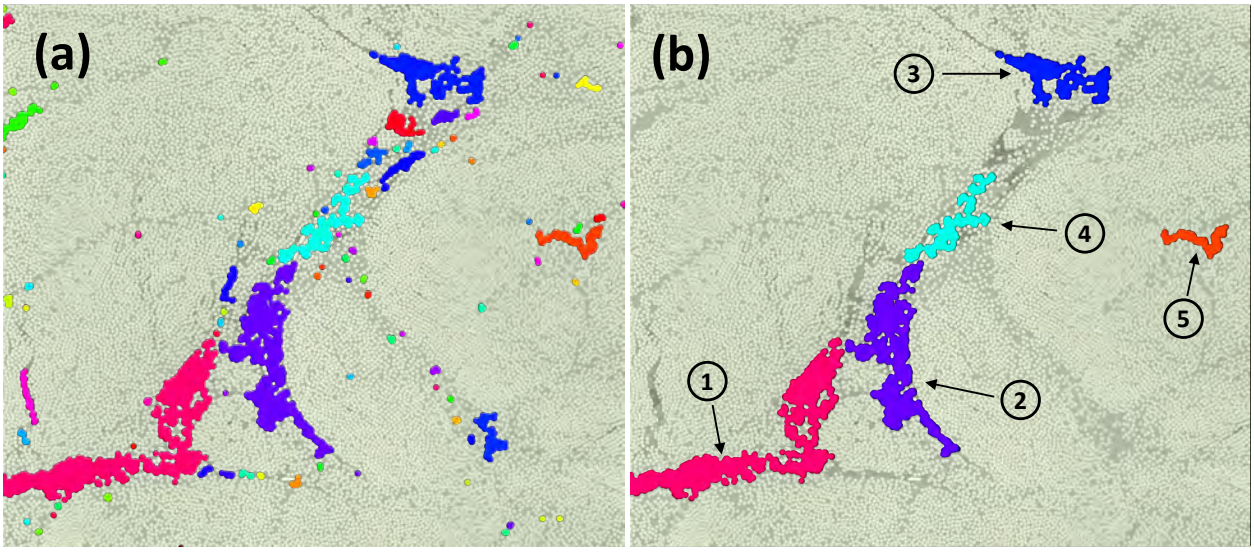

Figure 6.5: Discrete resin-rich areas are identified and labelled in different colors using our method. This method is able to output and visualize: (a) all detected resin-rich areas, or (b) the largest resin-rich areas (top 5 in this case). The test image is the same as shown in Fig. 6.1.

6.3.3 Selection of α value

It is critical to choose an appropriate α value because changing the α value changes the size and shape of the resin-rich areas. As illustrated in Fig. 6.6, a smaller α value means a smaller α-probe, which is able to freely move into smaller matrix regions and identify smaller resin-rich areas (Fig. 6.6(b)). When the α value increases, only significant matrix regions will be identified as resin-rich areas. The number of resin-rich areas decreases but the average resin-rich area size usually increases. A contiguous resin-rich area as identified using a small α value might be identified as multiple disconnected resin-rich areas using a large α value. For example, the middle light green resin-rich area in Fig. 6.6(b) is identified as multiple resin-rich areas under a larger α value in Fig. 6.6(c) because some narrow regions

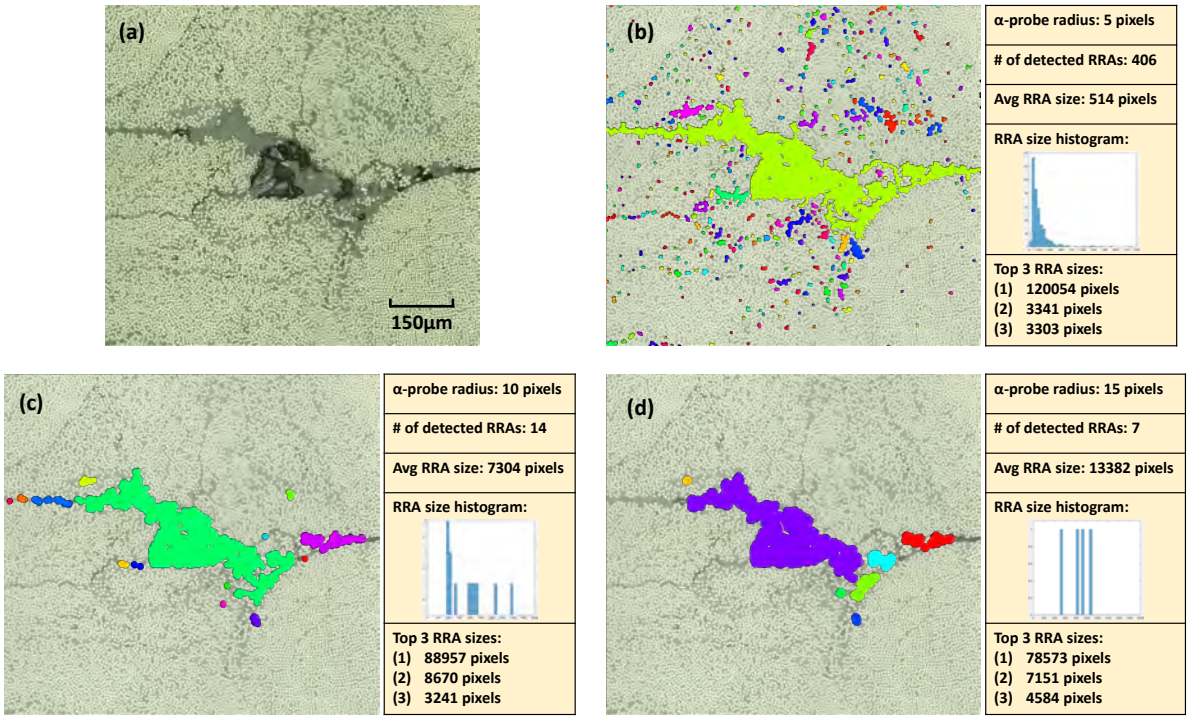

Figure 6.6: Changing the α value results in different detected resin-rich areas (RRAs). (a) input image, the nominal diameter of the fibers is 7 μm (10 pixels); (b) $\alpha = 5$ pixels (equals to the nominal radius R_n); (c) $\alpha = 10$ pixels ($2R_n$); (d) $\alpha = 15$ pixels ($3R_n$). In the figure, different colors indicate disconnected resin-rich areas.

in the light green resin-rich area are no longer accessible to the larger α-probe. If the α value is comparatively large, the algorithm will fail to detect narrow resin-rich areas, such as the resin seams.

The selection of the α value relates to the nominal radius (R_n, in pixels) of the fiber cross-sections. In our experiments, setting the α value between R_n and $3R_n$ generally provides useful resin-rich area detection results. Users are encouraged to select and test α values according to their research objectives. For example, Ghayoor et al. demonstrated that the failure strain in high volume fraction materials is sensitive to both large and small resin-rich areas [10]. For such research, a lower α value such as R_n is recommended to obtain thorough statistics of both large and small resin-rich areas. If the research aims to detect resin-rich seams/lines [16], a lower α value (R_n) is able to detect such narrow resin-rich patterns. On the other hand, if the research focuses on detecting and visualizing major resin-rich areas, a higher α value (such as $2.5R_n$ or $3R_n$) is recommended.

6.3.4 Algorithm efficiency

Our algorithm was implemented in MATLAB (version R2020a 9.8.0) and run on a laptop with an Intel® Core™ Processor i7-8550U CPU with 16GB RAM. In order to understand the run time of the algorithm with different input image sizes, square-shape testing images were cropped from the original microscope cross-sectional images. Fig. 6.7 shows the computation time of the algorithm for different test image sizes ranging from 1,000*1,000 to 10,000*10,000 pixels (α is set to R_n, 5 pixels).

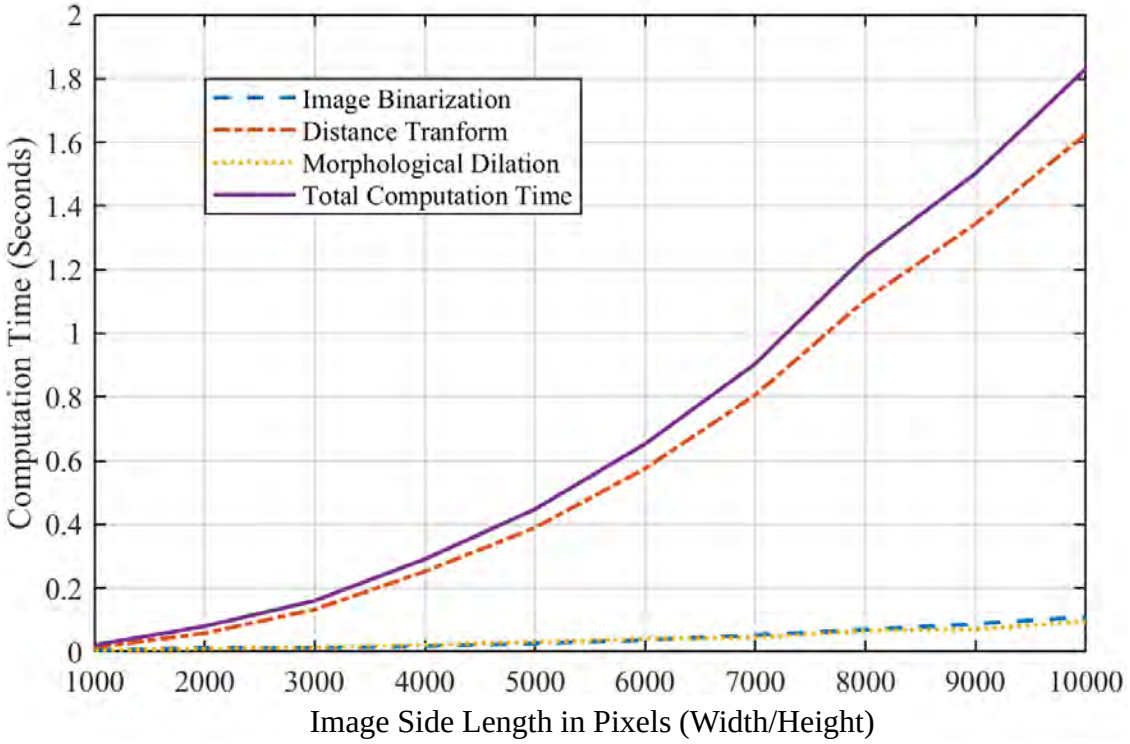

Figure 6.7: Computation time of the algorithm with different input image sizes. Square-shape images with sizes from 1000*1000 to 10000*10000 are tested. The α value is set as the nominal fiber radius R_n (5 pixels).

As shown in Fig. 6.7, all of these test examples are efficiently handled in less than 2 seconds. Consistent with our time complexity analysis (see B for details), the computation time of the whole algorithm or any of the three primary steps has a quadratic relationship to the side length (width or height) of the square-shaped test images. Regardless of image shape, the computation time of our algorithm has a linear relationship to the image area (width * height), so even for much larger images than those tested, the algorithm will run quickly. The average computation time on our experimental real-world full-size images (30 images with sizes around 18,000*10,000 pixels) is 3.38 seconds (ranging from 3.18 to 3.61

seconds). Calculating the distance transform is the most computationally expensive step in the algorithm, taking about 90% of the total computation time.

6.3.5 Comparison to prior approaches

For comparison, we implemented a Voronoi-based approach to resin-rich area detection [10, 22, 21]. These previous approaches require first determining the centers of fiber cross-sections; we do so using a more accurate watershed-segmentation based fiber recognition algorithm [18]. We then construct their Delaunay triangulation (the dual structure of the Voronoi diagram) and merge large-size triangles (those whose areas exceed a user-defined threshold) to form the detected resin-rich areas, as described in [10].

Experimental output of the resin-rich area detection on real-world inputs contrasting the Voronoi-based approach to our method is shown in Fig. 6.8. Compared to the Voronoi-based approaches, our method has advantages in terms of accuracy, efficiency, and applicability, as explained below.

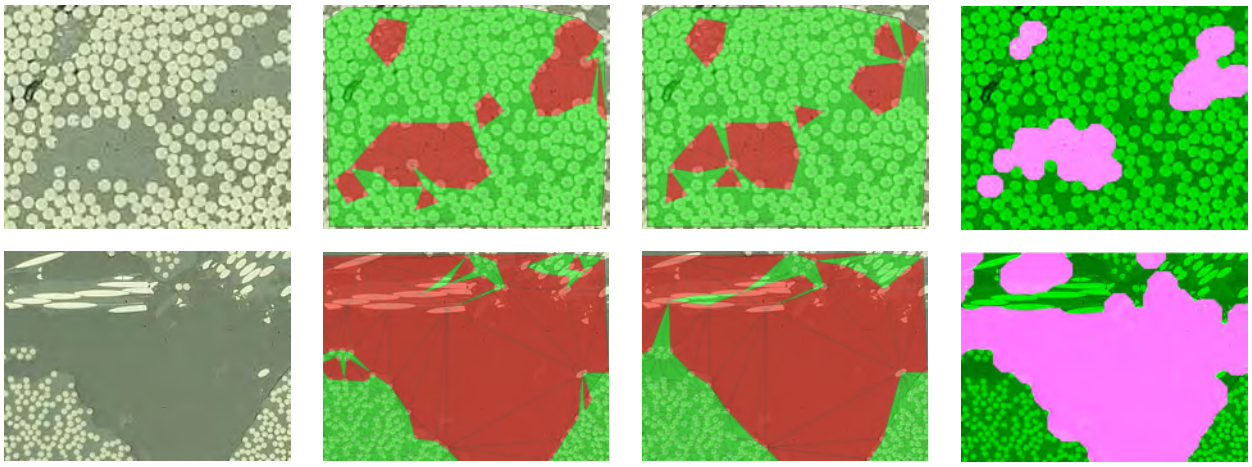

Figure 6.8: Experimental outputs of resin-rich area detection by the Voronoi-based approach and our method. From left to right: input images; Voronoi-based approach with a low threshold; Voronoi-based approach with a high threshold; our method.

Accuracy: One major shortcoming of using the Voronoi diagram concept to detect resin-rich areas is that although large-size Voronoi cells do indicate resin-rich areas, the reverse is not always true: resin-rich areas are not necessarily related to large-size cells in the Voronoi diagram, which means that small-size cells may also exist in the actual resin-rich areas. However, in the Voronoi-based approaches, such cells will be mis-classified as non-resin-rich areas, which leads to a mis-identification of large continuous resin-rich areas as series of smaller discrete resin-rich areas, regardless of parameter selection (Fig. 6.8 first row). By comparison, our method is able to accurately identify resin-rich areas in any situation.

Efficiency: on average, to detect resin-rich areas under each predefined parameter, the Voronoi-based approach takes 574 seconds per image. In comparison, our method only needs 3.46 seconds on average.

Applicability: Voronoi-based approaches assume equally-sized circular fiber cross-sections, which limits their applicability since fiber cross-sections can also appear as circles with varying sizes, ellipses, or even irregular shapes [17, 23, 24]. For example, Ahmadian et al. [17] state that instead of circular cross-sections of similar size, fibers in FRP composites in automotive applications have elliptical cross-sections whose sizes vary considerably. In comparison, our method is able to handle fiber cross-sections of arbitrary shapes and sizes, correctly calculating their resin-rich areas (Fig. 6.8 second row).

6.3.6 Void defect detection

In FRP composites, voids (or porosities) are also an important type of defect that has substantial influence on a wide range of material properties [66]. Voids appear as near-black regions in the FRP cross-sectional images, so they can be distinguished from fiber and matrix regions based on their pixel intensities. As an additional feature, along with resin-rich matrix areas, our implementation includes the option to differentiate void regions (Fig. 6.9). With this option, we modify the image binarization step (Section 6.2.1), applying Otsu's thresholding method [41] to automate segmenting input images into three levels (voids, fibers, and matrix) instead of two levels (fibers and non-fibers). In this three-level thresholding, Otsu's method automates determining the threshold values by minimizing the intra-class variance of the thresholded void, fiber, and matrix pixels. This void detection process takes an average of an additional 0.45 seconds per input image.

Since the major focus of this research is on the identification and quantification of structurally weak areas, although our implementation is also able to further distinguish void regions (from matrix resin-rich areas), we do not distinguish between them in other example figures throughout this paper.

6.4 Conclusions

This chapter presents a novel algorithm to automatically calculate resin-rich areas from composite cross-sectional images. Resin-rich areas are mathematically defined as the complementary areas of the fiber material's α-hull, and are efficiently calculated via the distance transform and morphological dilation operations. Detected resin-rich areas are further separated into disconnected regions from which a statistical summary can be derived. The proposed algorithm successfully handles real-world FRP cross-sectional images with fiber cross-sections of arbitrary shapes and sizes. The computation time of the algorithm has a linear relationship to the image area. The algorithm is able to calculate resin-rich areas from high-resolution microscope images (18,000*10,000 pixels), which contain more than 1.15 million fiber cross-sections, in less than 3.5 seconds. The detailed quantification and statistical

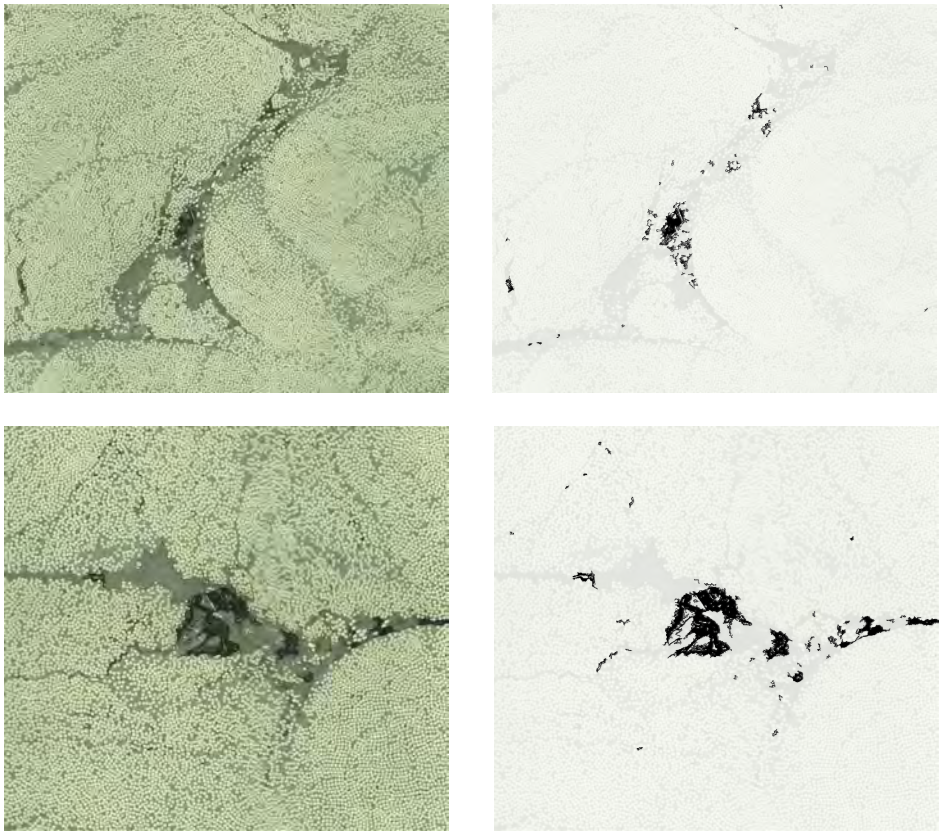

Figure 6.9: Input images (left) and void regions detected by our algorithm (right).

analysis of resin-rich areas enabled by this algorithm in turn opens up many future directions for related research in formulating the relationship between material properties and the sizes, locations, and morphologies of resin-rich areas.

Chapter 7

Discussion and Conclusions

7.1 Summary

In this dissertation, efficient and scalable geometric processing algorithms have been presented to automatically characterize the microstructure of additively manufactured FRP composites from their transverse microscope images. We validated our algorithms on more than 30 real-world microscope images that on average contain about 1.15 million fiber cross-sections. The algorithms successfully calculate the sizes, locations, and breakage of fiber cross-sections, and identify the resin-rich area defects with high accuracy and efficiency. The major contributions in this dissertation are:

- We propose the first fiber recognition algorithm that is able to automatically detect and categorize fiber cross-sections as aligned, misaligned, or broken fibers, from cross-sectional microscope images of FRP composites.
  - We exploit distance-transform-based watershed segmentation to identify individual fiber cross-sections.
  - We introduce a new geometric tool, contour gradient charts (CGC) to identify broken fibers as well their unbroken contours, for more accurate categorization and localization of individual fiber cross-sections.
  - We validate our algorithm on real-world microscope images from 3D-printed FRP parts, on which our method is able to correctly identify and categorize more than 99.9% of fibers.
- We design a sample-based algorithm to construct Voronoi diagrams of 2D circles or 3D spheres.
  - The algorithm successfully handles input circles/spheres in both general and non-general position.

  - We design a GPU framework to exploit data parallelism for efficiency improvement. Compared to the exact computation of Voronoi diagrams, the total run time of the resin-rich area detection process is reduced by over 50% by integrating this Voronoi construction algorithm.

- We develop two methods to automatically define and characterize resin-rich areas from microscope images, based on the concept of α-shape/hull.
  - We apply the concept of α-shape/hull to formulate a mathematical definition of the boundaries of resin-rich areas. To our knowledge, this is the first approach to quantitatively summarize resin-rich areas.
  - We design exact computation and sample-based algorithms to calculate the α-shape and α-hull, respectively. The overall computation time of the sample-based algorithm is 3.5 seconds for real-world testing images that contain more than 1.15 million fiber cross-sections on average.
  - The rigorous mathematical definition of resin-rich areas and ability to collect thorough statistics will facilitate better understanding and quantification of the relationship between resin-rich areas and material properties.

7.2 Comparison between two defect detection approaches

In Chapter 4 and 6, two methods (Voronoi approach and distance transform approach) are introduced to mathematically define and calculate resin-rich areas in FRP composites. Although both successfully handle real-world examples and outperform previous approaches, these two methods utilize different methodologies to calculate the alpha-shape/alpha-hull, which lead to different advantages and disadvantages. We now compare different characteristics of the two methods in terms of accuracy, efficiency, and applicability (summarized in Fig. 7.1).

Accuracy: Both our Voronoi and distance transform approaches are able to robustly and accurately identify resin-rich areas. However, the Voronoi approach can provide more meaningful results because during the fiber recognition pre-processing step, broken and misaligned fibers can be distinguished from aligned fibers. This allows users to have the choice to remove broken and misaligned fibers from consideration in the resin-rich area calculation, since they are usually considered to be defects in composite materials. In comparison, the distance transform approach does not distinguish between broken, aligned, and/or misaligned fibers.

Efficiency: The distance transform approach is more computationally efficient. The time complexity (as well as the run time) of the distance transform approach is dependent only on the size of the input image (pixels), regardless of the shapes or sizes of fiber cross-sections. For example, the algorithm is able to process a high-resolution 18,000*10,000

	Voronoi Approach	Distance Transform Approach
Process	Fiber Recognition → Voronoi → Dual → α-shape	Image Thresholding → Distance Transform → Morphology → α-hull
Space Domain	Continuous	Discrete
Accuracy	√ +	√
Efficiency		√
Extensibility		
• **Composite Laminate**	√	
• **Arbitrary Fiber Shape**		√
• **3D Extension**		√

Figure 7.1: Comparison between the Voronoi approach and the distance transform approach.

microscope image in 3.5 seconds. In comparison, the Voronoi approach requires 74.5 seconds (excluding the fiber recognition process) to process the same image because of the time-consuming computations in continuous space. Moreover, the Voronoi approach is sensitive to the unpredictable number of misaligned fibers, so the run time may further increase when more misaligned fibers exist.

Extensibility: Beyond detecting defects of unidirectional FRP materials, the Voronoi approach is also well-suited for the analysis of other continuous-FRP-based structures such as laminate structures or woven structures. In these continuous-FRP-based structures, fibers appear in bundles [21], within which the fiber cross-sections have similar orientation/ellipticity. The fiber recognition step enables the Voronoi approach to identify these fiber bundles by clustering fibers with the same orientations, transform each of the identified fiber bundles to its transverse orientation, and detect its resin-rich areas accordingly. On the other hand, the distance transform approach is able to analyze FRP composites with not only circular and elliptical fiber cross-sections, but also arbitrary/irregular fiber cross-section shapes. The algorithm could also be extended to detect resin-rich areas in 3D space by utilizing the 3D versions of distance transform and morphological operations (which are available in MATLAB).

7.3 Future work

In this research, we focus on transverse cross-sectional images of unidirectional continuous FRP composites. Future work could include extending our methods to other continuous-FRP-based structures and short-fiber reinforced polymers. As discussed above, for continuous-

FRP-based structures, it is feasible to detect their resin-rich areas by extending the Voronoi approach with a new bundle recognition step. For short-fiber reinforced polymers, 2D cross-sectional images do not provide sufficient information for understanding their microstructures. Extending the current distance transform approach from 2D to 3D, employing 3D α-hulls, one could then detect resin-rich areas directly on 3D scans of short-fiber reinforced polymers.

The mathematical definition of resin-rich areas by α-shapes/hulls will facilitate better understanding and quantification of the relationship between resin-rich areas and material properties. One intriguing direction to explore is using the detailed quantitative characterization of resin-rich areas as input features for machine learning models that could predict material strength, volume resistivity, toughness, or failure strain. For example, the geometric distribution of the resin-rich areas could form the basis for shape descriptors [67] to be used as input features in machine learning models. Alternatively, the resin-rich area size histograms (shown in Fig. 6.6) can be treated as input features in histogram-based machine learning models such as [68].

In 3D printed FRP materials, large resin-rich areas usually indicate gap areas between printing strips (also called beads) or layers. One area for future research could be automatically recognizing boundaries between strips (as well as the separate individual strips) by connecting appropriate detected resin-rich areas (Fig. 7.2). Recognition of individual strips will enable us to better evaluate the manufacturing process by determining the amount of in-strip vs. inter-strip defects. For instance, substantial in-strip defects usually indicate imperfections in the raw material, whereas substantial inter-strip defects usually indicate an excessive bead width in the 3DP setup.

In conclusion, our automated algorithms for characterizing additively manufactured FRP composites assist in inspection of material microstructures, facilitate quantitative prediction of material properties, and support optimization of 3D printing process parameters. We hope that our algorithms and tools become an integral part of future research in composite materials characterization, and industrial automation for additive manufacturing of composites.

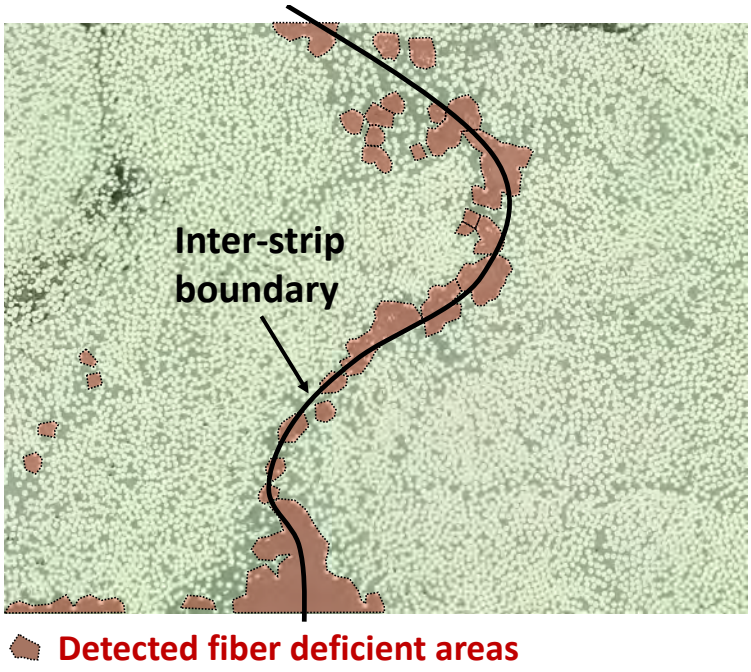

Figure 7.2: Adjoining resin-rich areas along an inter-strip boundary.

Bibliography

[1] Chuncheng Yang, Xiaoyong Tian, Tengfei Liu, Yi Cao, and Dichen Li. "3D printing for continuous fiber reinforced thermoplastic composites: mechanism and performance." In: *Rapid Prototyping Journal* 23.1 (2017), pp. 209–215.

[2] Frank Van Der Klift, Yoichiro Koga, Akira Todoroki, Masahito Ueda, Yoshiyasu Hirano, and Ryosuke Matsuzaki. "3D printing of continuous carbon fibre reinforced thermo-plastic (CFRTP) tensile test specimens." In: *Open J. Compos. Mater* 6.1 (2016), pp. 18–27.

[3] Grand View Research. *Composites Market Size, Share & Trends Analysis Report*. July 2020. URL: `https://www.grandviewresearch.com/industry-analysis/composites-market`.

[4] J Zhang, BL Fox, D Gao, and AW Stevenson. "Inspection of drop-weight impact damage in woven CFRP laminates fabricated by different processes." In: *Journal of composite materials* 43.19 (2009), pp. 1939–1946.

[5] Pedram Parandoush, Chi Zhou, and Dong Lin. "3D printing of ultrahigh strength continuous carbon fiber composites." In: *Advanced Engineering Materials* 21.2 (2019), p. 1800622.

[6] Kentaro Sugiyama, Ryosuke Matsuzaki, Andrei V Malakhov, Alexander N Polilov, Masahito Ueda, Akira Todoroki, and Yoshiyasu Hirano. "3D printing of optimized composites with variable fiber volume fraction and stiffness using continuous fiber." In: *Composites Science and Technology* 186 (2020), p. 107905.

[7] WJ Cantwell and J Morton. "The significance of damage and defects and their detection in composite materials: a review." In: *The journal of strain analysis for engineering design* 27.1 (1992), pp. 29–42.

[8] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. "NIH Image to ImageJ: 25 years of image analysis." In: *Nature methods* 9.7 (2012), p. 671.

[9] Sivasankaran Harish, D Peter Michael, A Bensely, D Mohan Lal, and A Rajadurai. "Mechanical property evaluation of natural fiber coir composite." In: *Materials characterization* 60.1 (2009), pp. 44–49.

[10] Hossein Ghayoor, Catharine C Marsden, Suong V Hoa, and António R Melro. "Numerical analysis of resin-rich areas and their effects on failure initiation of composites." In: *Composites Part A: Applied Science and Manufacturing* 117 (2019), pp. 125–133.

[11] Brian S Hayes and Luther M Gammon. *Optical microscopy of fiber-reinforced composites*. ASM international, 2010.

[12] Bent F Sørensen and Ramesh Talreja. "Effects of nonuniformity of fiber distribution on thermally-induced residual stresses and cracking in ceramic matrix composites." In: *Mechanics of materials* 16.4 (1993), pp. 351–363.

[13] VN Bulsara, Ramesh Talreja, and J Qu. "Damage initiation under transverse loading of unidirectional composites with arbitrarily distributed fibers." In: *Composites science and technology* 59.5 (1999), pp. 673–682.

[14] Masaki Hojo, Masaaki Mizuno, Thomas Hobbiebrunken, Taiji Adachi, Mototsugu Tanaka, and Sung Kyu Ha. "Effect of fiber array irregularities on microscopic interfacial normal stress states of transversely loaded UD-CFRP from viewpoint of failure initiation." In: *Composites Science and Technology* 69.11-12 (2009), pp. 1726–1734.

[15] Shinichiro Yamashita, Yuto Nakashima, Jun Takahashi, Kazumasa Kawabe, and Tetsuhiko Murakami. "Volume resistivity of ultra-thin chopped carbon fiber tape reinforced thermoplastics." In: *Composites Part A: Applied Science and Manufacturing* 90 (2016), pp. 598–605.

[16] Francisco Sacchetti, Wouter JB Grouve, Laurent L Warnet, and Irene Fernandez Villegas. "Effect of resin-rich bond line thickness and fibre migration on the toughness of unidirectional Carbon/PEEK joints." In: *Composites Part A: Applied Science and Manufacturing* 109 (2018), pp. 197–206.

[17] Hossein Ahmadian, Ming Yang, and Soheil Soghrati. "Effect of resin-rich zones on the failure response of carbon fiber reinforced polymers." In: *International Journal of Solids and Structures* 188 (2020), pp. 74–87.

[18] Xiang Li, Sara Shonkwiler, and Sara McMains. "Fiber recognition in composite materials." In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 2623–2627.

[19] Xiang Li, Adarsh Krishnamurthy, Iddo Hanniel, and Sara McMains. "Edge topology construction of Voronoi diagrams of spheres in non-general position." In: *Computers & Graphics* 82 (2019), pp. 332–342.

[20] Xiang Li, Sara Shonkwiler, and Sara McMains. "Detection of resin-rich areas for statistical analysis of fiber-reinforced polymer composites." In: *Composites Part B: Engineering* 225 (2021), p. 109252.

[21] F Gommer, A Endruweit, and AC Long. "Quantification of micro-scale variability in fibre bundles." In: *Composites Part A: Applied Science and Manufacturing* 87 (2016), pp. 131–137.

[22] Heechun Yang and Jonathan S Colton. "Quantitative image processing analysis of composite materials." In: *Polymer Composites* 15.1 (1994), pp. 46–54.

[23] Qingping Sun, Haiding Guo, Guowei Zhou, Zhaoxu Meng, Zhangxing Chen, Hongtae Kang, Sinan Keten, and Xuming Su. "Experimental and computational analysis of failure mechanisms in unidirectional carbon fiber reinforced polymer laminates under longitudinal compression loading." In: *Composite Structures* 203 (2018), pp. 335–348.

[24] Gyu Jeong, Jae Hyuk Lim, Chunghyeon Choi, and Sun-Won Kim. "A virtual experimental approach to evaluate transverse damage behavior of a unidirectional composite considering noncircular fiber cross-sections." In: *Composite Structures* 228 (2019), p. 111369.

[25] Fausto Bernardini, Chandrajit L Bajaj, Jindong Chen, and Daniel R Schikore. "Automatic reconstruction of 3D CAD models from digital scans." In: *International Journal of Computational Geometry & Applications* 9.04n05 (1999), pp. 327–369.

[26] Laurent-Philippe Albou, Benjamin Schwarz, Olivier Poch, Jean Marie Wurtz, and Dino Moras. "Defining and characterizing protein surface using alpha shapes." In: *Proteins: Structure, Function, and Bioinformatics* 76.1 (2009), pp. 1–12.

[27] MATLAB. *version 9.8.0 (R2020a)*. Natick, Massachusetts: The MathWorks Inc., 2020.

[28] The CGAL Project. *CGAL User and Reference Manual*. 5.2.1. CGAL Editorial Board, 2021. URL: `https://doc.cgal.org/5.2.1/Manual/packages.html`.

[29] Jae-Kwan Kim, Youngsong Cho, Donguk Kim, and Deok-Soo Kim. "Voronoi diagrams, quasi-triangulations, and beta-complexes for disks in R2: The theory and implementation in BetaConcept." In: *Journal of Computational Design and Engineering* 1.2 (2014), pp. 79–87.

[30] Deok-Soo Kim, Jeongyeon Seo, Donguk Kim, Joonghyun Ryu, and Cheol-Hyung Cho. "Three-dimensional beta shapes." In: *Computer-Aided Design* 38.11 (2006), pp. 1179–1191.

[31] Deok-Soo Kim, Donguk Kim, and Kokichi Sugihara. "Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry." In: *Computer Aided Geometric Design* 18.6 (2001), pp. 563–585.

[32] Ioannis Z Emiris and George M Tzoumas. "Exact and efficient evaluation of the InCircle predicate for parametric ellipses and smooth convex objects." In: *Computer-Aided Design* 40.6 (2008), pp. 691–700.

[33] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. "On the shape of a set of points in the plane." In: *IEEE Transactions on information theory* 29.4 (1983), pp. 551–559.

[34] Herbert Edelsbrunner and Ernst P Mücke. "Three-dimensional alpha shapes." In: *ACM Transactions on Graphics (TOG)* 13.1 (1994), pp. 43–72.

[35] Ricardo Fabbri, Luciano Da F Costa, Julio C Torelli, and Odemir M Bruno. "2D Euclidean distance transform algorithms: A comparative survey." In: *ACM Computing Surveys (CSUR)* 40.1 (2008), pp. 1–44.

[36] B Mlekusch. "Fibre orientation in short-fibre-reinforced thermoplastics II. Quantitative measurements by image analysis." In: *Composites Science and Technology* 59.4 (1999), pp. 547–560.

[37] Julio Martin-Herrero and Ch Germain. "Microstructure reconstruction of fibrous C/C composites from X-ray microtomography." In: *Carbon* 45.6 (2007), pp. 1242–1253.

[38] Sung Joon Ahn, Wolfgang Rauh, and Hans-Jürgen Warnecke. "Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola." In: *Pattern Recognition* 34.12 (2001), pp. 2283–2303.

[39] K Amjad, WJR Christian, K Dvurecenska, MG Chapman, MD Uchic, CP Przybyla, and EA Patterson. "Computationally efficient method of tracking fibres in composite materials using digital image correlation." In: *Composites Part A: Applied Science and Manufacturing* 129 (2020), p. 105683.

[40] Yonghong Xie and Qiang Ji. "A new efficient ellipse detection method." In: *Object recognition supported by user interaction for service robots*. Vol. 2. IEEE. 2002, pp. 957–960.

[41] Nobuyuki Otsu. "A threshold selection method from gray-level histograms." In: *IEEE transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66.

[42] Calvin R Maurer, Rensheng Qi, and Vijay Raghavan. "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.2 (2003), pp. 265–270.

[43] Fernand Meyer. "Topographic distance and watershed lines." In: *Signal Processing* 38.1 (1994), pp. 113–125.

[44] Andrew Fitzgibbon, Maurizio Pilu, and Robert B Fisher. "Direct least square fitting of ellipses." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.5 (1999), pp. 476–480.

[45] Tim J Atherton and Darren J Kerbyson. "Size invariant circle detection." In: *Image and Vision Computing* 17.11 (1999), pp. 795–803.

[46] Deok-Soo Kim, Donguk Kim, and Kokichi Sugihara. "Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology." In: *Computer Aided Geometric Design* 18.6 (2001), pp. 541–562.

[47] Li Jin, Donguk Kim, Lisen Mu, Deok-Soo Kim, and Shi-Min Hu. "A sweepline algorithm for Euclidean Voronoi diagram of circles." In: *Computer-Aided Design* 38.3 (2006), pp. 260–272.

[48] Mokwon Lee, Kokichi Sugihara, and Deok-Soo Kim. "Topology-oriented incremental algorithm for the robust construction of the Voronoi diagrams of disks." In: *ACM Transactions on Mathematical Software (TOMS)* 43.2 (2016), pp. 1–23.

[49] Donguk Kim and Deok-Soo Kim. "Region-expansion for the Voronoi diagram of 3D spheres." In: *Computer-aided design* 38.5 (2006), pp. 417–430.

[50] Zhongyin Hu, Xiang Li, Adarsh Krishnamurthy, Iddo Hanniel, and Sara McMains. "Voronoi cells of non-general position spheres using the GPU." In: *Computer-Aided Design and Applications* 14.5 (2017), pp. 572–581.

[51] Ioannis Z Emiris, Elias P Tsigaridas, and George M Tzoumas. "Exact Voronoi diagram of smooth convex pseudo-circles: General predicates, and implementation for ellipses." In: *Computer Aided Geometric Design* 30.8 (2013), pp. 760–777.

[52] Deok-Soo Kim, Youngsong Cho, Kokichi Sugihara, Joonghyun Ryu, and Donguk Kim. "Three-dimensional beta-shapes and beta-complexes via quasi-triangulation." In: *Computer-Aided Design* 42.10 (2010), pp. 911–929.

[53] Jonathan Richard Shewchuk. "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator." In: *Workshop on Applied Computational Geometry*. Springer. 1996, pp. 203–222.

[54] Xiaoyu Zheng and Peter Palffy-Muhoray. "Distance of closest approach of two arbitrary hard ellipses in two dimensions." In: *Physical Review E* 75.6 (2007), p. 061709.

[55] Iddo Hanniel and Gershon Elber. "Computing the Voronoi cells of planes, spheres and cylinders in R^3." In: *CAGD* 26.6 (2009), pp. 695–710.

[56] Gershon Elber and Myung-Soo Kim. "Computing rational bisectors." In: *IEEE Computer Graphics and Applications* 19.6 (1999), pp. 76–81.

[57] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm." In: *ACM SIGGRAPH*. Vol. 21. 4. ACM. 1987, pp. 163–169.

[58] *The RCSB Protein Data Bank*. `http://www.rcsb.org/pdb/home/home.do`. Accessed: 2015-12-14.

[59] Marina L Gavrilova and Jon Rokne. "Updating the topology of the dynamic Voronoi diagram for spheres in Euclidean d-dimensional space." In: *CAGD* 20.4 (2003), pp. 231–242.

[60] Kokichi Sugihara. "Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams." In: *CVGIP: Graphical Models and Image Processing* 55.6 (1993), pp. 522–531.

[61] Anil K Jain and Marie-Pierre Dubuisson. "Segmentation of X-ray and C-scan images of fiber reinforced composite materials." In: *Pattern Recognition* 25.3 (1992), pp. 257–270.

[62] Julian Besag. "On the statistical analysis of dirty pictures." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 48.3 (1986), pp. 259–279.

[63] Tarik Arici, Salih Dikbas, and Yucel Altunbasak. "A histogram modification framework and its application for image contrast enhancement." In: *IEEE Transactions on image processing* 18.9 (2009), pp. 1921–1935.

[64] Mohammad Abdullah-Al-Wadud, Md Hasanul Kabir, M Ali Akber Dewan, and Oksam Chae. "A dynamic histogram equalization for image contrast enhancement." In: *IEEE Transactions on Consumer Electronics* 53.2 (2007), pp. 593–600.

[65] D. Zhang, N. Rudolph, and P. Woytowitz. "Reliable optimized structures with high performance continuous fiber thermoplastic composites from additive manufacturing (AM)." In: *International SAMPE Technical Conference*. 2019.

[66] Mahoor Mehdikhani, Larissa Gorbatikh, Ignaas Verpoest, and Stepan V Lomov. "Voids in fiber-reinforced polymer composites: A review on their formation, characteristics, and effects on mechanical performance." In: *Journal of Composite Materials* 53.12 (2019), pp. 1579–1669.

[67] Ismail Khalid Kazmi, Lihua You, and Jian Jun Zhang. "A survey of 2D and 3D shape descriptors." In: *2013 10th International Conference Computer Graphics, Imaging and Visualization*. IEEE. 2013, pp. 1–10. DOI: 10.1109/CGIV.2013.11.

[68] Amalya Zeynalova, Burak Kocak, Emine Sebnem Durmaz, Nil Comunoglu, Kerem Ozcan, Gamze Ozcan, Okan Turk, Necmettin Tanriover, Naci Kocer, Osman Kizilkilic, et al. "Preoperative evaluation of tumour consistency in pituitary macroadenomas: a machine learning-based histogram analysis on conventional T2-weighted MRI." In: *Neuroradiology* 61.7 (2019), pp. 767–774. DOI: 10.1007/s00234-019-02211-2.

[69] Marcel Van Herk. "A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels." In: *Pattern Recognition Letters* 13.7 (1992), pp. 517–521.

[70] Xinhua Zhuang and Robert M Haralick. "Morphological structuring element decomposition." In: *Computer vision, graphics, and image processing* 35.3 (1986), pp. 370–382.

Appendix A

Comparison between Distance Transform Approach and Pixel-wise Approximation

A possible alternative (approximate, pixel-wise) approach would be to identify resin-rich areas by applying the concept of α-shapes. Similar to α-hulls, α-shapes can be used to approximate the areas that can not be accessed by the α-probe. Since the computation of α-shapes from point set inputs is available in many software packages, a variation of our algorithm could detect resin-rich areas by approximating each fiber pixel (identified as described in Section 6.2.1) as an input point, building the α-shape of such input points using available software packages such as MATLAB or CGAL (replacing Section 6.2.2 and 6.2.3), and computing its complement. We implemented this alternative pixel-wise approximation method building on the alphaShape() function in MATLAB. However, compared to our proposed α-hull method, this pixel-wise approximation method is orders of magnitude slower (Table A.1). For example, for the real-world high-resolution image of 18,270*10,306 pixels it took over 85 minutes, while our proposed linear-time α-hull method takes less than 3.5 seconds. Furthermore, the computation time in Table A.1 is for each single choice of α. In practice, users may wish to explore different α values to find the one most suited to their research objectives or to collect additional statistics.

Image size (pixels)	**Computation time (pixel-wise)**	**Computation time (ours)**
2000*2000	14.28s	0.08s
4000*4000	59.56s	0.29s
6000*6000	161.43s	0.65s
8000*8000	463.16s	1.24s
10000*10000	2429.20s	1.83s
18270*10306 (real-world size)	5116.26s (85.27min)	3.46s

Table A.1: Computation time comparison (for each choice of α) between the pixel-wise approximation and our proposed α-hull method.

Appendix B

Time Complexity Analysis: α-hull Construction via Distance Transform

Here we demonstrate that the computation time of our algorithm in practice has a linear relationship to the area (n by m pixels) of the input image.

There are three major steps in our algorithm: image binarization, distance transform calculation, and morphological dilation calculation. Given an input image of size n by m pixels and probe radius of α, the run-time complexity of the three steps are as follows.

In the image binarization step, Otsu's method [41] is implemented, which has a time complexity $MAX(\mathcal{O}(nm), \mathcal{O}({N_{bins}}^2))$, where N_{bins} is the number of bins in the image histogram, usually set as 256. Because the cross-sectional images for FRPs usually have a

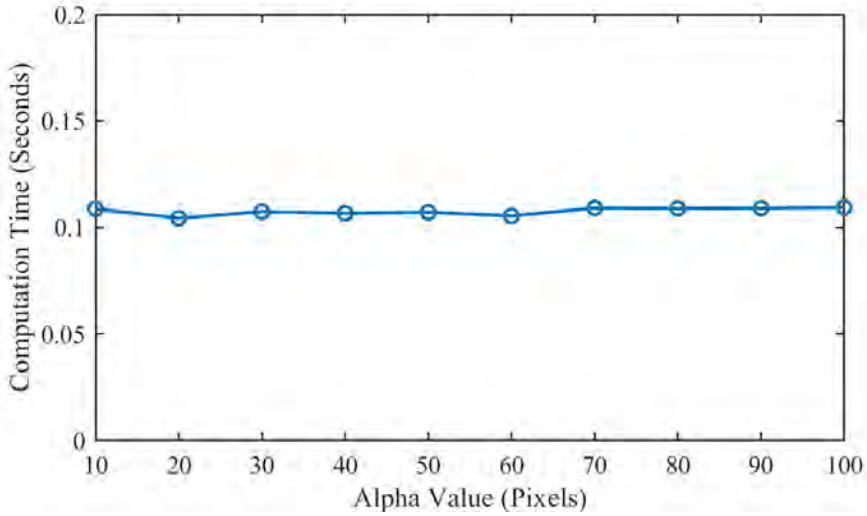

Figure B.1: Computation time of the morphological dilation operation with different α values. The input image size is 5,000*5,000 pixels.

high resolution (m, n much greater than 256), the complexity of image binarization can be considered as $\mathcal{O}(nm)$.

In the distance transform operation, the algorithm proposed by Maurer et al. [42] is applied. It is a linear time algorithm with regards to the size (number of pixels) of the input image. So the run time complexity of this step is also $\mathcal{O}(nm)$.

In the morphological dilation operation, the naive implementation has a time complexity $\mathcal{O}(nm\alpha^2)$, relating to both the size of the input image and the probe size α. However, in our implementation by using optimization methods [69, 70] and due to the fact α is much smaller than n and m, the morphological dilation operation time is nearly independent of the α value (Fig. B.1). So the run time of this step is also linear to (nm) in practice, only depending on the size of the input images.

Since the first two steps in the algorithm have time complexity $\mathcal{O}(nm)$, and the third step is also linear to (nm) in practice, the overall computation time of our algorithm in practice has a linear relationship to the size of the input image (nm).