Implicit Representation of Inscribed Volumes

by

Parto Sahbaei
B.Sc., Payame Noor University, 2010

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Implicit Representation of Inscribed Volumes

by

Parto Sahbaei
B.Sc., Payame Noor University, 2010

Supervisory Committee

Dr. Brian Wyvill, Co-Supervisor
(Department of Computer Science)

Dr. David Mould, Co-Supervisor
(Department of Computer Science, Carleton University)

Dr. Andrea Tagliasachi, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Brian Wyvill, Co-Supervisor
(Department of Computer Science)

Dr. David Mould, Co-Supervisor
(Department of Computer Science, Carleton University)

Dr. Andrea Tagliasachi, Departmental Member
(Department of Computer Science)

ABSTRACT

We present an implicit approach for constructing smooth isolated or interconnected 3-D inscribed volumes which can be employed for volumetric modeling of various kinds of spongy or porous structures, such as volcanic rocks, pumice stones, *Cancellus bones*[*], liquid or dry foam, *radiolarians*, cheese, and other similar materials. The inscribed volumes can be represented in their normal or positive forms to model natural pebbles or pearls, or in their inverted or negative forms to be used in porous structures, but regardless of their types, their smoothness and sizes are controlled by the user without losing the consistency of the shapes. We introduce two techniques for blending and creating interconnections between these inscribed volumes to achieve a great flexibility to adapt our approach to different types of porous structures, whether they are regular or irregular. We begin with a set of convex polytopes such as 3-D Voronoi diagram cells and compute inscribed volumes bounded by the cells. The cells can be irregular in shape, scale, and topology, and this irregularity transfers to the inscribed volumes, producing natural-looking spongy structures. Describing the inscribed volumes with implicit functions gives us a freedom to exploit volumetric surface combinations and deformations operations effortlessly.

[*]For the definition, refer to Glossary section of this document.

Contents

List of Figures

Acronyms

CSG Constructive Solid Geometry

LoD Level of Details

CAD Computer Aided Designs

MLS Moving Least Squares

RBF Radial Basis Function

MPU Multi-level Partition of Unity

CSB Constructive Solid Blobs

MC Marching Cubes

MT Marching Tetra-hedra

EMC Extended Marching Cubes

DMC Dual Marching Cubes

DC Dual Contouring

RGBA Red, Green, Blue, and Alpha or opacity

NURBS Non-Uniform Rational Basis Spline

IC Inscribed Curves

IV Inscribed Volumes

SSS Subsurface Scattering

ACKNOWLEDGEMENTS

I would like to thank:

my supervisors Dr. Brian Wyvill and Dr. David Mould, whose expertise, guidance, support, and patience throughout my graduation experience steered me in the right direction to finish my thesis work. I am truly grateful for giving me the opportunity of working with you.

my mother, father, and sister for their support, motivation, inspiration and love through my entire life.

my dear husband, Shaham, whose love, patience, encouragements and editing assistance has helped me along the way.

my colleague Herbert Grasberger for his assistance and sharing ideas during my master studies.

DEDICATION

I dedicate this work to my parents and my husband for all their support.

Chapter 1

Introduction

We introduce a general method for volume modeling of spongy or porous surfaces including radiolarians, pumice stones, volcanic rocks, cheese, dry foams, and similar volumes. Proposing a generic method is still a challenging task because of the complex geometry or surfaces of these substances, as mentioned by Bear et al. [6]. Most current approaches are restricted to handle some particular types or aspects of these complex models or with the focus on the surface material properties. One reason it is difficult to formulate a general and versatile approach, arises from the morphological and structural distinctions between the above examples. Porous materials, such as volcanic rocks or cheese, are volumes with pores that may or may not interconnect. Sponges are specific types of porous materials that have full bodies of pores which means they have an internal skeleton of *spongin*. Modeling of such objects is not only a matter of surface material and therefore to achieve general representation of them, volumetric surface modeling is required.

We aim to model a general form of porous structures that are not only natural looking but aesthetically satisfying concerning the smoothness of their surfaces. Our approach consists of an improved basic 3-D method derived from an earlier 2-D result by Wyvill et al [1], empowered by new two interconnection techniques to accomplish its versatility in various applications. The interconnections happen between inscribed volumes closed by some convex boundaries, however, the style of these connections, regularity and chaotic considerations is what makes these two techniques difference from each other and adaptable for many porous structures. The inscribed volumes (IV) in their positive form can be used to model natural pebbles, liquid foams, or similar objects, however, to achieve complex models, such as spongy structure models, they should be defined in a negative form by hollowing out other solid primitives

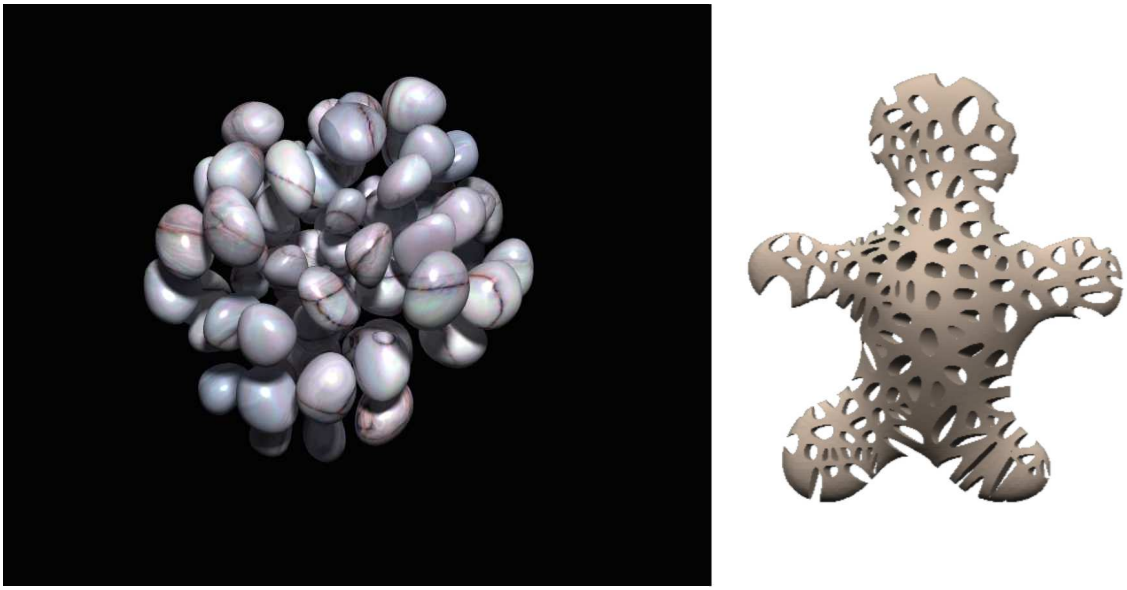


Figure 1.1: Inscribed Volumes in positive form (left) and negative form (right).

using Constructive Solid Geometry (CSG) or a negative blending operation (Figure 1.1). The framework of this research employs other techniques to fulfill some specific aspects in constructing IVs and spongy models, namely implicit volumes modeling and 3-D Voronoi.

Volumetric surface modeling, whether for the purpose of scientific visualization or entertainment, has matured as primary field of computer graphics in the recent years. Shirley et al. [5] describe a wide range of techniques for representing geometry to construct or reconstruct, model, edit and render 3-D volume surfaces. In general, these representations fall into two major categories of parametric and implicit representations [7], where both are used to specify the geometric shape of an object.

Implicit volumes modeling [5] produces smooth and aesthetic shapes that can be seamlessly modified at the local or global level while keeping consistent modeling structure to produce target results [3,8]. In addition, their function definition is compact and require less high-level primitives to construct a model [9]. Wyvill et al. [1] conducted a user survey to specify which class of curves, whether parametric or implicit, are acknowledged to be more appealing for human perception. The results of this user study showed a remarkable preference of implicit curves over B-spline curves, which is mathematically a more appropriate approach because of continuous mapping from convex polygons, as opposed to the continuous mapping of a sequence of points in B-spline, to the curves. Apart from the aesthetic aspects, we needed an intuitive

modeling structure to construct, combine, and alter simple primitives without explicit or post-process mesh modifications. Moreover, implicit surfaces provide some unique characteristics that qualify them as a better practical alternative to parametric surfaces. Their abilities for easy bending, warping, deforming [10], and blending [11, 12] as well as performing Constructive Solid Geometry (CSG) operations [13], are just a few to name. A hierarchical modeling structure containing primitives and operations, i.e. the BlobTree by Wyvill et al. [8], is employed in our framework which gives freedom in producing complicated models with inscribed volumes, for example, performing 3-D noise displacements on negative IVs to construct naturally shaped rock.

Our approach uses 3-D Voronoi cells to produce convex, stochastic, and natural looking geometry with the ability to control the topology of the inscribed volumes and infer the adjacency of neighboring cells. The Voronoi diagram is a robust tool for representing many geometric structures for partitioning space into convex polytopes, respectively. Furthermore, the Voronoi diagram plays a key role in many applications such as modern art, decorative and architectural design, and natural science, duo to its capability to produce aesthetic and natural patterns [14–17].

1.1 Contributions

Our main contributions to this research are:

- The construction of 3-D inscribed volumes using an implicit modeling approach.
- The introduction of two new interconnection techniques for inscribed volumes in a 3-D Voronoi diagram.
- The introduction of a new distance field facilitating the editing of the inscribed volumes.

1.2 Structure

The structure of this work is as follows: in Chapter 1, we describe the problems in modeling porous materials and introduce our general methodology and its important concepts; in Chapter 2, we provide a literature review of previous work on porous structures, bubbles, implicit modeling and rendering, and aesthetic studies on 2-D

and 3-D curves; Chapter 3 contains a more detailed technical background of the basic concepts that are the building blocks of this research contributions; in Chapter 4, we describe our main methodology for constructing inscribed volumes and two new generalization techniques to increase the versatility of our approach; in Chapter 5, the results and discussions on our framework capabilities to generate different porous or similar structures and a comparison of it with current related word are provided; finally in Chapter 6, we provide a conclusion and describe some possible future work. In Appendix A, the Implementation details of a Dual Contouring polygonization technique has provided for those interested in implementing our method.

Chapter 2

Literature Review

2.1 Porous Structures

Work has been done for constructing and visualizing volumetric spongy structures where some are based on constructing symmetric spongy volumes, some considered the volumetric reconstruction of 3-D textures, and others inspired by the open cell 3-D Voronoi. Symmetric 3-D structure representations, such as the Menger sponge-like fractal body [18], do not provide a general and smooth method to satisfy a stochastic structure of natural sponges.

For surface modeling and rendering of objects with porous or *mesoscopic* details, some texture synthesis approaches were suggested. Tong et al. [19] proposed a bidirectional texture function synthesis considering both surface geometry and surface details to achieve realistic porous materials. The synthesis is done by examining distances between 2-D surface *texton*s derived from 3-D texton method [20]. The drawback with textons is that the clustering introduces discretization errors, and the distance metric requires costly access to a large inner-product matrix which makes this method computationally slow [21]. This method also used copy-based synthesis to generate a full-scale pore on the surface which was not arbitrarily chosen. Later, they proposed Quasi-Homogeneous materials [22] which produced more realistic surface materials with a complex capture and rendering process. These methods are basically focused on the representing porous surface materials with 2-D texture synthesis and rendering techniques on objects such as bread with outer shells mesoscopic pores.

Chen et al. [23] studied 3-D texture mapping which uses microstructures to model

internal structures for the purpose of product design and manufacturing. The proposed texture mapping approach is based on mapping a 3-D microstructure pattern of *truss* geometry into a design space to generate internal cellular structures which then are combined with a given Computer Aided Design (CAD) model of the object. This representation scheme can be used to define general structures and build a library of microstructures for modeling *Biomimetic* design, e.g. bones, cartilages, and honeycombs, and layer manufacturing, e.g. *stereolithography* and selective laser sintering. The 3-D microstructure they used for their 3-D texture mapping were regular geometry shapes such as tetrahedron, cube, dodecahedron, triacontahedron, and similar shapes. The final structures made with these regular geometries could be warped to some extents and are more suited for the industrial purposes and less beneficial for natural spongy structures due to lack of required stochastic properties.

Magda et al. [24] described a method for acquiring volumetric surface textures which separate geometric information from the reflectance data. In this method, the pixel wise surface normal field, local reflectance functions, and light attenuation parameters are computed for individual layers separately which leads to a significant modeling effort. Since their algorithm offers high compression of the original data, it is especially suitable for the textures with complex, opaque microstructure, shadowing, and unknown reflectance properties that are difficult to compress, and it can also offer fully rendered textures at any orientation to the viewer at extra memory and computation cost. This method produces both regular and random 3-D textures structures used for rendering 3-D details including pores on an arbitrary polygonal surface.

Baravalle et al. [25] presented an interesting method for modeling porous structures by a particle system generation of 3-D textures using a dynamical system. Their method is a 2-D sliced based texture mapping which means that the 3-D textures apply to a specific slice of an object which helps with computational efficiency of the method. Their framework system supports sufficient randomness of size and orientation of pores required for photo-realistic rendering and modeling of many porous structures. This method does not produce smooth or round pores in some structures, such as radiolarians, cheese, or spongy bones, however, is able to represent mesostructure details of bread or similar textures on the outer shell of a surface.

For modeling bubbles, Zheng et al. [26] introduced a general method to simulate realistic dynamic effects of bubbles cluster. In this work, the implicit surface of the liquid film can be adjusted to achieve an arbitrary thickness of the bubble walls.

They have used a regional level set method which allows representing a thin film as the boundary between two gas regions. In this method, each bubble is implicitly associated with a region identifier and a level set function that samples distance to the bubble boundary. This helps to track multi-manifold surfaces for thin bubble films with a semi-implicit surface tension model and compute the shape of bubbles efficiently. Their technique has the ability to produce large bubbles where the deformation of the bubbles is important for visual effects.

Another method is introduced by Busaryev et al. [27] where a particle-based algorithm for simulating small bubbles and foams is considered. Their algorithm is based on approximating the foam geometry by considering bubble particles as the sites of a weighted 3-D Voronoi diagram. By having the connectivity information provided by the 3-D Voronoi diagram, accurate bubble dynamics and interactions among bubbles can be modeled. This approach is capable of simulating realistic looking small wet and dry foams, however, it has surface deformation restrictions, like other particle-based approaches, and is not suitable for large bubble simulations.

Modeling open-cells foam is a known method for representing dry foams with hollowed internal structures. Martinez et al. [28] proposed a method for modeling and studying different properties (i.e. elasticity, regularity, and randomness) of foam microstructures in sufficiently large scale object inspired by Voronoi open-cells foams. The microstructures in this method are directly generated to exhibit a specified elastic behavior required for industrial purposes. Using implicit modeling for their technique allows the generating of the very detailed structures in large objects without having to produce a full representation, such as mesh or voxels of the complete object explicitly in a seamless manner. This approach is beneficial and optimized for the standards of industrial manufacturing of synthetic foams, nevertheless, it is less appealing for modeling of naturally smooth or random spongy structures.

Due to the complex structure of porous materials, it is very difficult to determine the relationship between their structure and properties. The complex functional behavior of foams can be studied by an exact characterization of their microstructure followed by numeric simulations. Several methods have been developed to build numerical models of such materials. One way is fitting a stochastic geometric model to the microstructure, based on the characteristics measured in a 3-D image. Subsequently, changing the model parameters alters the microstructure. After some iterations, the microstructure can be optimized with respect to the desired porous materials property. Some of the frequently used stochastic models for this method are Voronoi and

Laguerre tessellations for foams. In some methods, a microstructure model based on the combination of 3-D image analysis and random *Laguerre tessellation*s is presented which captures the basic features of real solid foams based on foam skeleton [29, 30]. A similar cell foam modeling method based on Voronoi tessellations is also presented by Miliaris et al. [31]. These techniques have the same issues as mentioned for the Martinez method [28] and they can be used for specific structures and applications and were not intended as a generic method for other types of spongy structures.

2.2 Implicit Modeling

Implicit modeling gained attentions and interests of many scholars due to an intuitive representation of surfaces in a mathematical way, rather than explicit geometric representation. Ricci [3] was one of the pioneers for defining and representing implicit surfaces, where he also employed CSG for surface modeling in computer graphics. The modeling with filter fall-off function was attributed to Jim Blinn [5] when he introduced *Blobby Molecules* modeled after electron density fields [32]. In his model, the functions are designed to locally present the surface and have a free-form style, meaning that the defined surfaces can move around freely. Other popular earlier work on filter fall-off functions were presented by Nishimura et al. for introducing *Metaballs* [33] and *Soft objects* by Wyvill et al. in 1986 [34]. Since then, numerous methods were proposed [4, 35–38]. Therefore, implicit modeling found its way in different applications from scientific visualizations and medical data illustrations to computer aided designs (CAD), animation, computer games, and more [39–41].

Skeleton-based implicit functions are a generalization of Blobby models, Metaballs, and Soft Objects in that instead of using point-based modeling by summation of only point potentials, other basic primitives such as lines, planar curves and regions, polygons, and in principle any geometric models are practiced. Bloomenthal et al. [42] introduced *Convolution Surfaces* in 1991 as a powerful re-interpretation and generalization of potential surfaces by using convolution with piecewise planar skeletons to model a surface. Other attractive aspects of their method were smooth blending and the ability for easy blending of potential fields of different primitives with bulge elimination properties. Wyvill et al. [8, 11] introduced *The BlobTree* as an underlying data structure and extending skeletal implicit modeling for generating complex shapes consisting of an arbitrary number of primitives, CSG operations, and warping at global and local levels. Schmidt [43] presented an interactive implicit

modeling technique with hierarchical spatial caching using the BlobTree to accelerate evaluations of the potential functions. This caching method is able to reduce the number of tree nodes traversed at each single point by storing the exact potential fields at the nodes of a 3-D uniform grid. Grasberger et al. [44] proposed another extension for modeling *fillet* with the BlobTree, known as *Constructive Solid Blobs* (CSB). Their method supports shapes that are different from the standard CSG approach and creates smooth transitions between hard-edged and soft-edged primitives in a Boolean operation. A few 3-D implicit modeling and design tools software packages were developed by employing skeletal-based modeling, such as BlobTree.NET [45] by De Groot and ShapeShop [39] by Schmidt. The Shapeshop software uses a sketched-based shape creation and editing tools which allow users to produce easy, fast, smooth, and seamless objects by using drag-and-drop interactions.

A breakthrough in constructive implicit modeling was proposed by Gourmel et al. [4] by introducing gradient-based blending which solves several major issues in previous composition operators. Their precise method, particularly in bulge elimination, was employed in Implicit Skinning [46], as studied by Vaillant et al.

2.2.1 Rendering

Fast and precise visualization of implicit surfaces is computationally intensive. An overview and comparison of existing implicit modeling and rendering techniques are provided in the literature [47–49]. Some well-known techniques of rendering constructive implicit modeling in 3-D space are ray-tracing [50], polygonization [51], and direct volume rendering [52]. The two former approaches regarded as the most famous and well-used methods for their high quality rendering and versatility features, respectively.

Ray-tracing is a technique for generating a direct sampling image of a surface at its exact intersection points with a ray vector. Appel in 1968 [53] proposed the first ray-casting algorithm used for rendering and Roth [54] was the first who employed ray-casting algorithm in computer graphics. The further work on ray-tracing were presented to improve the accuracy and performance of rendered scenes [55–60]. Despite creating high quality rendered images, ray-tracing can be quite slow especially when not optimized. To guarantee of finding all surface intersection and to speed up the evaluations, many techniques were suggested, such as using the *Lipschitz Constant* as proposed by Kalra et al. [56] to rapidly discard non-intersecting rays with voxels

using octree [58] or kd-tree [61]. De Groot et al. [62] introduced *Rayskip* method for fast ray-tracing of implicit surfaces animations by reducing the number of evaluating rays and using space and temporal coherence.

Polygonization is an alternative hardware-friendly, easy to render, versatile, and appealing approach since it is able to extract a mesh of polygons by approximating a volumetric representation. With polygonization, interactive real-time rendering is possible and the final mesh is portable and can be reproduced on many platforms and altered according to different applications' needs. A recent survey on fast implicit polygonization techniques was accomplished by Araujo [47], and a fast method for subdividing a 3-D space into a grid of cells (i.e. voxels) for polygonization was proposed by Wyvill et al. [34]. A year later, the *Marching Cubes* (MC) algorithm was introduced by Lorensen et al. in 1987 [63]. In this algorithm, intersections of surface models with voxel edges are calculated and a mesh of triangles is created by connecting the intersected points.

The main difference between MC and Wyvill method [34] is that in MC every voxel is stored but the Soft Objects method stores only surface intersecting voxels. The original MC was slow due to evaluating all voxels in space and failed to identify some ambiguous cases, while the Wyvill polygonization algorithm had the second set of voxels to improve the efficiency by only considering overlapping primitives and identified the ambiguity problem as opposed to MC. Some other popular polygonization techniques are *Marching Tetrahedra* (MT) [9, 64, 65], *Extended Marching Cubes* (EMC) [66], *Dual Marching Cubes* (DMC) [67], and *Dual Contouring* (DC) [68]. Jepp et al. [69] presented a particle system method for rendering complex implicit objects which was able to represent shape features including sharp edges.

2.3 Aesthetic Aspect

In favor of the aesthetic appeal of 2D and 3D curves and curvatures, several studies have been accomplished. This aesthetical appeal of planar curves is usually called fairness in industrial design [70]. Fairness in curves can be obtained when the curvature changes smoothly or piece-wise monotonically [71, 72], where each piece may have linear curvature changes. Farin et al. [70] and McCrae [73] adjusted parametric curves to increase fairness. Wyvill et al. [1], proposed using implicit approach for constructing patterns formed by closed curves rather than explicit curves. Wong et al. [74] articulated the important principles of order, repetition, balance, and confor-

mation to constraints as influencing the appeal of a pattern. In evaluating aesthetic curves several work considered various algorithms and factors to determine fairness or aesthetic curve and curvature [75–78]. There is no doubt that no algorithm is fully capable of judging aesthetics with the reliability and granularity to replace human judgments and in this work, we relied on the result of a user survey accomplished by Wyvill et al. [1] where 2-D inscribed implicit curves received more preference votes compared to parametric (B-Spline) curves.

2.4 Voronoi Diagram

Voronoi Diagram is a computational technique to investigate many geometric structures by partitioning a 3-D Euclidian space or a 2-D plane into convex volumes or polygons, respectively. Many pioneer works have introduced Voronoi diagrams in various applications and studied accelerating its construction. Shamos et al. [79] and Fortune [80] introduced a divide-and-conquer algorithm and a sweep-line algorithm to construct Voronoi. A simple method to construct a Voronoi diagram of line segments sites by incrementally expanding the segments and using kinematic methods to maintain the Voronoi diagram was also suggested by Gold et al. [81]. Later, Ledoux [82] proposed a general algorithm for 3-D Voronoi construction regardless of the spatial distribution of the input points. Hsieh et al. [83] introduced an efficient and simple method for Voronoi construction, however, without any constraints for the model. An efficient and versatile 3-D method to compute a clipped cell-based Voronoi tessellation, where features of Voronoi cells (e.g. volume, centroid, the number of faces) can be used to analyze a system of particles was developed by Rycroft [84, 85]. Their method supports radial Voronoi tessellation and is able to provide fast neighboring computations and a custom complex boundary for any number of control points in 3-D space.

Chapter 3

Technical Background

This chapter deals with the main technical backgrounds on inscribed curves and mathematical fundamentals of implicit surfaces concepts which are required for the next chapters. The inscribed curves method [1] is the 2-D foundation of our proposed methods. As it was explained in the previous chapter there are several major techniques for modeling implicit surfaces, in this chapter we particularly focus on skeletal implicit modeling because of their compactness and intuitiveness for flexibility in manipulations and the fact that they are a main part of the BlobTree. The BlobTree does not directly influence the definition of inscribed volumes described in Chapter 4, nevertheless, its usage in our framework gives us the freedom to have an intuitive and flexible data structure to construct desired models with them.

3.1 Inscribed Curves

The idea behind inscribed curves (IC) in 2-D method [1], is to define a set of *aesthetically pleasing* curves in the interior of each bounded convex polygon of a 2-D Voronoi diagram in a way that they approximate their corresponding region. In their context, the aesthetic aspect refers to smoothness properties of the closed curves and they defined two types of inscribed curves functions: parametric curves and implicit curves. The two approaches were used to investigate and compare their aesthetic appeals. In both approaches, they defined each curve in an individual convex polygon, so its definition was independent of the actual Voronoi diagram. For each convex polygon $\mathbb{P}$ the following data is available: a vertex set consists of $\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_{k-1}$; $\mathbf{n}_i$ as the inward unit normal for l_i, where l_i is a edge connecting $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$.

In the implicit curve approach, the signed distance of an arbitrary point $\mathbf{x} \in \Re^2$ to the edge l_i is

$$d_i(\mathbf{x}) = (\mathbf{x} - \mathbf{p}_i) \cdot \mathbf{n}_i, \tag{3.1}$$

where the $(\cdot)$ denotes dot product.

For each point located on the edge l_i we have

$$l_i = \left\{ \mathbf{x} \in \Re^2 \mid d_i(\mathbf{x}) = 0 \right\}. \tag{3.2}$$

The goal was to define a function F that was positive inside $\mathbb{P}$ and unbounded (undefined) on the boundaries of $\mathbb{P}$. The value of F outside $\mathbb{P}$ is not a matter of importance since the IC, as the name suggests, is going to be defined on the interior of $\mathbb{P}$, therefore it is desired to be unbounded outside of $\mathbb{P}$.

Let S_a be the points on the surface as $S_a = \left\{ \mathbf{x} \,|\, F(\mathbf{x}) = a \right\}$ and $F(\mathbf{x})$ be the implicit function with positive level-set (iso-lines) values of a, then

$$F(\mathbf{x}) = \prod_{0 \le i < k} d_i(\mathbf{x}), \tag{3.3}$$

where k is the total number of vertices of $\mathbb{P}$. According to (3.3), any point positioned on any edge of $\mathbb{P}$ will result to $F(\mathbf{x}) = 0$. If the defined function is convex and smooth over the interior of $\mathbb{P}$, the iso-curve of this function would be convex as well. Function F in equation (3.3) is defined over $\Re^2$. To assure the convexity of equation (3.3), function f can be defined as

$$f(\mathbf{x}) = \log F(\mathbf{x}), \quad \mathbf{x} \in \mathbb{P}. \tag{3.4}$$

Considering 3.3 and 3.4, we have

$$-f(\mathbf{x}) = -\log \prod_{0 \le i < k} d_i(\mathbf{x}) = -\sum_{0 \le i < k} \log\, d_i(\mathbf{x}), \quad \mathbf{x} \in \mathbb{P}. \tag{3.5}$$

An issue with (3.5) happens when a tiny sized edge adds to the polygon. Since all edges have the same influence in this equation, a small edge can effectively change the shape of the curve. To fix this issue, they added an edge length weight to the function f, thus each edge could change the shape regarding of its size, therefore the

equation changes to

$$f(\mathbf{x}) = \sum_{0 \leq i < k} \lambda_i \, \log d_i(\mathbf{x}), \tag{3.6}$$

where λ_i indicates the *i-th* edge length.

The function f which is defined over a convex polygon with positive real values is now *logarithmically convex*, as the $-\log(.)$ is convex and the summation of convex functions is convex. The smoothness and geometric deformation of inscribed iso-curves depend on the value of a, namely if a approaches to zero then S_a provides a closer approximation of polygon $\mathbb{P}$ boundaries and the larger the value of a, the rounder and smoother iso-curves will be produced. The authors also claimed that the function f can be defined with any other qualified convex functions other than logarithm, as long as the function is convex, positive inside and unbounded on the boundary and outsides of the region.

For producing parametric curves, they used a degree n B-spline basis approach (the higher the degree, the smoother the curve) where the vertex points of the convex polygon $\mathbb{P}$ were the controls points of the B-spline curves and can locally affect the shape of curves. To guarantee that the curve is closed and is bounded by the convex hull of polygon $\mathbb{P}$, they repeated the first point of control points at the end of the sequence. They defined the function of the curve as

$$c(t) = \sum_{i=0}^{k+n} \mathbf{p}_{(i \bmod k+1)} N_i^n(t), \quad t \in [0, \; k+n], \tag{3.7}$$

where $N_i^n(t)$ is the basis function computed by *DeBoor Cox* algorithm [86].

The spline curve is quite sensitive to the addition of a tiny edge to the polygon and can remarkably reduce the continuity of the curve near the control points of that edge. A solution for this issue can be tricky as the spline curve defined by the control points whereas the implicit curves take the edges weights into consideration. It is possible to add weighting for knot values or control points, nevertheless, this solution does not effectively diminish the influence of one of two points of a tiny edge, hence the smoothness cannot be guaranteed (see Wyvill [1]).

To verify the connection between smoothness and aesthetic appeal of a curve for human perception, the authors conducted a user-study. The results showed a remarkable preference of implicit curves over the parametric curves, which were expected as the B-spline curves were C^2 continuous and there were computed as the mapping

from control points whereas the implicit curves were C^{∞} continuous and were computed as a continuous map from the convex polygon. Figure 3.1 illustrates a snapshot example of comparison (among 30 pairs) in their user-study survey, as it can be seen each user were asked to provide his/her opinion about each pair of ICs.

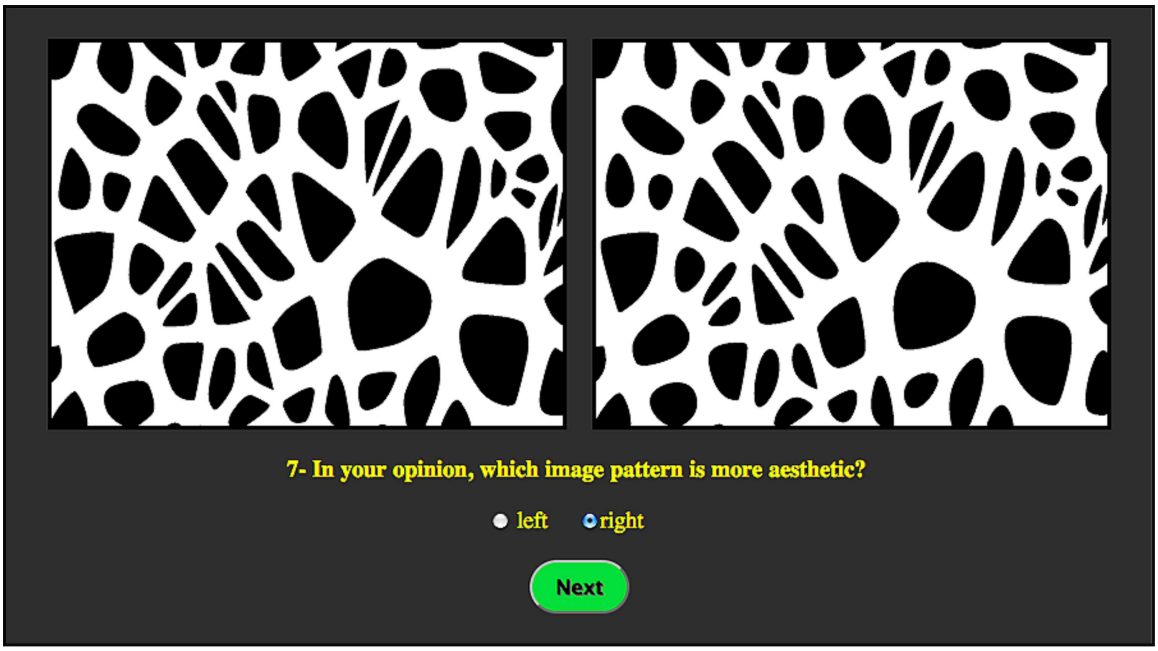


Figure 3.1: An example of user-study comparison of parametric curves (left) and implicit curves (right) from Wyvill et al. [1].

3.2 Skeleton-Based Implicit Surfaces

In general, an implicit surface is defined as the set of points in Euclidean space where a function $f : \Re^3 \rightarrow \Re$ is equals to a constant real value a.

$$f(x, y, z) = a, \tag{3.8}$$

where a is called *iso-value*. The reason of why they are called *implicit*, is simply because without solving the equation there is no direct way to know if a point $\mathbf{p} \in \Re^3$ lies on the surface or not. For zero value surfaces (i.e. $a = 0$), if the value of $f(\mathbf{p})$ is positive, it means that the point $\mathbf{p}$ is outside of the surface, if $f(\mathbf{p})$ is negative, $\mathbf{p}$ is inside, and finally if $f(\mathbf{p})$ is zero, $\mathbf{p}$ is located on the surface.

Skeleton-based implicit surfaces was proposed as an expansion to Blobby Molecules,

Meta-balls, and Soft Objects by utilizing more varieties of geometries to model an object rather than solely usage of point skeletons (see Figure 3.2). Blobby Molecules had an infinite domain and therefore when the number of control primitives increases it becomes computationally expensive to evaluate the contributions of all primitive at every point in space. In addition, having unbounded support makes the blending of two primitives less precise and harder to control. On the other hand, Meta-balls and Soft Objects have a bounded region and thus provide better performances. What all of these functions have in common is that their field values at each point in $\Re^3$ are derived from the distance of that point to each control primitive(s) and as a result is continuous. If the control primitive is a non-point object, such as a line or circle, the distance value will be the shortest distance of a point in space to the control primitive. The evaluated distance value is semi-definite as its value is bigger or equal to zero, however, a distance filter function can be used to bound its domain as well as the derived field values.

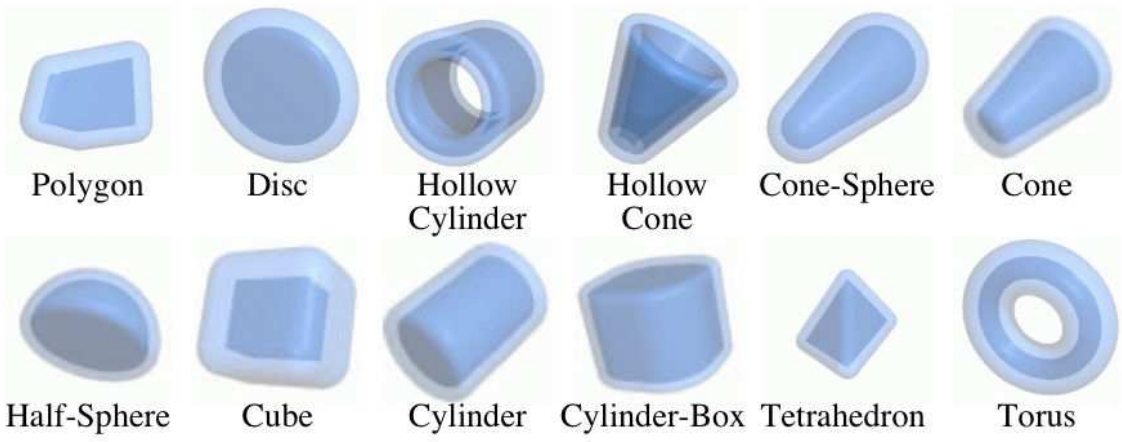


Figure 3.2: Some examples of skeletal primitives, image from Barbier et al. [2].

3.3 The BlobTree

Perhaps the greatest strength of skeletal implicit surfaces is their ability to blend smoothly, nevertheless, in different applications other ways of combining primitives would be desired. The combination of finite number of primitives helps to achieve complex shapes and there are a few well-known methods to produce them, such as (1) Constructive Solid Geometry (CSG) which uses the basic Boolean operations of union,

intersection, and difference between half space primitives and they may be performed by a hierarchical Construction Tree (2) different blending approaches such as: Ricci, locally restricted, bounded, and gradient-based blending (3) adequate inner bound for implicit surfaces, a modified version of gradient-blending which reduces blending artifacts (4) the BlobTree, as a hierarchical modeling framework.

The BlobTree, as mentioned earlier, is a hierarchical modeling method which allows users a freedom to compose arbitrary shapes by using binary, unary, and n-ary operations in a homogeneous manner. The basic binary operations are similar to CSG tree Boolean operations, n-ary such as blending, and the typical unary operations are Barr's space warping [13] which includes: bending, twisting, taper, and essentially any deformation on a surface along the tree. The leaves of the tree are always implicit primitives and middle nodes up to the root are any of the mentioned operations. One of the advantages of BlobTree over CSG tree is its the ability to perform space warping both globally and locally, e.g. a twist operation can be applied over one specific leaf primitive, one parts the tree, or the whole tree. Figure 3.3 illustrates a simple example of the BlobTree.

To query the field value or normal at each point in space, the BlobTree may be traversed. The field value (or normal vector) for point $\mathbf{p}$ is extracted by traversing from the bottom leaf nodes up to the root of the tree and its value at each node N along the tree is denoted as $F(N)$. For interactive frameworks, the value of $F(N)$ can be cached to avoid expensive computational costs for those objects that are not meant to be modified. The traversal of BlobTree is explained in the Appendix A, and the different operations that are implemented for this project are detailed as follows.

In this tree, most shapes are built from skeletal primitives where the scalar field value at each point is taken from the unsigned distance value from the skeleton, an appropriate fall-off filter function, such as the Wyvill function [5], is then applied to provide C^2 continuous model. For all of the objects defined in this work, the field value for the middle or centroid of an object is always equal to *one*, and it decreases towards the boundaries of the object, i.e. radius or bounded region, with the minimum value of *zero*. This enables a huge computational saving at any point that is farther enough from the skeleton and therefore it is not necessary to be evaluated. An arbitrary iso-value a is then picked, where this value defines the desired iso-surface boundary and separates the outer and inner space. When locally supported field functions are desired, the iso-value of $a = 0.5$ is usually defined.

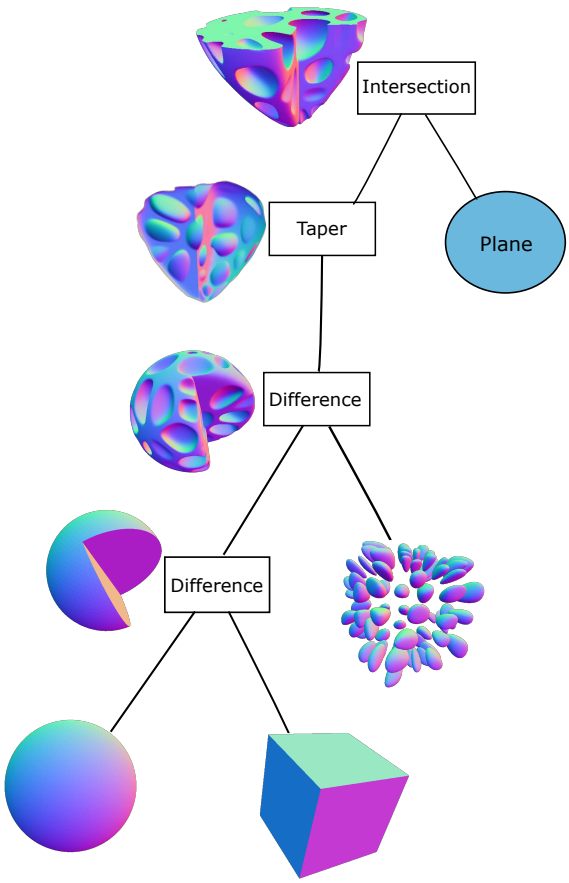


Figure 3.3: A Simple Example of the BlobTree.

3.3.1 Blending, Binary and Unary Operations

The simplest blending operation between two or more objects can be performed by the summation of their potentials at any point in a bounding volume in 3-D space:

$$g(f_1, f_2) = f_1 + f_2, \tag{3.9}$$

where f_1 and f_2 are the field values of two object at point $\mathbf{p} \in \Re^3$ and g is the *summation blend* operator, as mentioned by Bloomenthal [7]. Nevertheless, using the (3.9) blending function, there will be less control on the final shape as it suffers from topology and bulging problems. A simple blending function that can be applied to local support field functions i.e. fall-off to *zero*, is super-elliptic blending, aka *Ricci* Blending [3]:

$$g(f_1, f_2) = \left(f_1^t + f_2^t\right)^{1/t} \tag{3.10}$$

The main difference between (3.9) and (3.10) is t which enables it to have a better control over the sharpness and overall shapes of blending objects. When t converges to $+\infty$, the blending behaves similar to maximum or union operator, and likewise, when it converges to $-\infty$, the blending will be close to intersections of the input objects (Figure 3.4). Using the equation (3.10) has another advantage over summation as it makes the interpolation between blending and union operation easier (i.e. by changing the parameter t). A low value of parameter t will result in unwanted shape changes at the joints, and a high value will result in loosing smooth transitions between two objects.

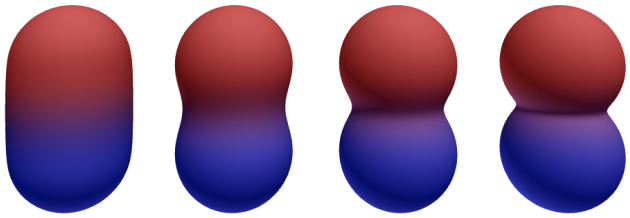


Figure 3.4: Ricci Blending [3], from left to right, the parameter t is equal to 1, 2, 4 and 8. Note that for $t = 1$ the blending is the same as equation (3.9)

CSG binary operations are performed on scalar field values of two primitives. A hierarchical data structures of CSG operations thanks to Construction Tree and the BlobTree, is also possible. These operations on their simplest forms are as follows:

$$\left\{ \begin{array}{ll} (f_1 \cup f_2)(\mathbf{p}) = max\left(f_1(\mathbf{p}), f_2(\mathbf{p})\right) & \text{, union} \\ (f_1 \cap f_2)(\mathbf{p}) = min\left(f_1(\mathbf{p}), f_2(\mathbf{p})\right) & \text{, intersection} \\ (f_1 - f_2)(\mathbf{p}) = min\left(f_1(\mathbf{p}), 1 - f_2(\mathbf{p})\right) & \text{, difference} \end{array} \right\}$$

The main issue with the above equations is that they generate C^0 surfaces, and according to the application this may not be desired, however, their versatile straightforward definitions make them as one of the first choices in CSG operations. To overcome the discontinuity issues, a few C^n equations such as *clean union* operation [87] and [12], have been proposed. Although there are several other *clean union* [87] and blending operators defined for globally support field functions [7], we skip them since the skeletal and bounded locally supports implicit surfaces are the main concerns of

this project. The restrictions and properties of a locally supported function makes finding well-behaved composition operators harder than global support. In general, an ideal composition operator according to Gourmel et al. [4] must overcome these issues: (1) bulging problem specially at the joints (2) locality problem which results in blending at distance (3) absorption problem or blurring of details (4) topology problem, e.g. when a hole in the middle of a surface gets filled as a result of blending. Bernhardt et al. [12] proposed a solution based on considering an extra implicit primitive and using its field value to interpolate between blending and union, however having only G^1 continuity the resulting surfaces may suffer from curvature artifacts. Gourmel et al. [4] introduced gradient-based blending which solves the aforementioned four shortcomings effectively. The solution to this problem was to devise composition functions based on both gradient and distance fields. (see Figure 3.5).

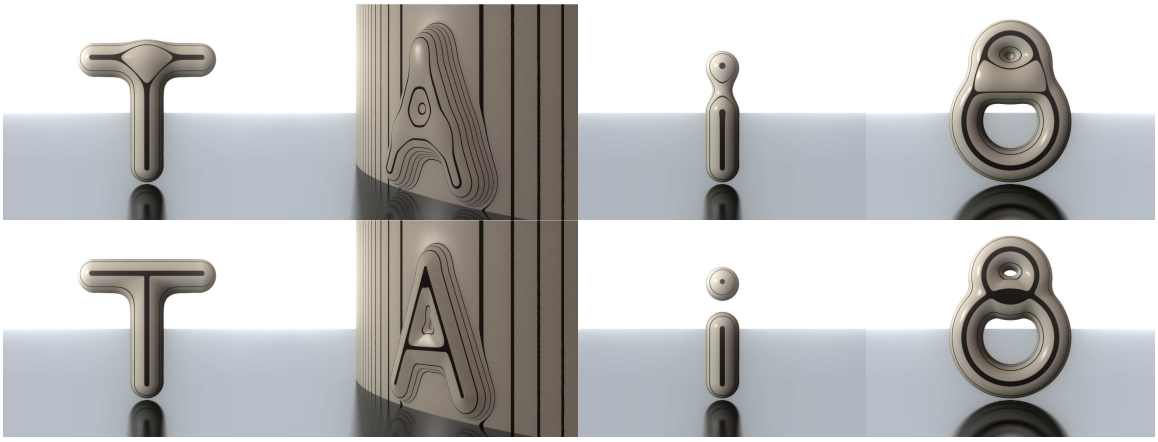


Figure 3.5: Comparison of standard blending (top row) four main issues, and proposed solution by Gourmel et al. [4] gradient-based blending operator (bottom row), the issues from left to right: unwanted bulge, blow up of small details, blending at distance, and topology modification (Image from [4]).

Warp operation is a group of deformations of rigid objects which is based on transforming a shape within its hull as the hull gets deformed. The basic space-warp operations introduced by Barr [13] (aka Barr's Deformations) are: twisting, bending, and tapering. Figure 3.6 shows two blended cylinders after using these operations in the BlobTree.

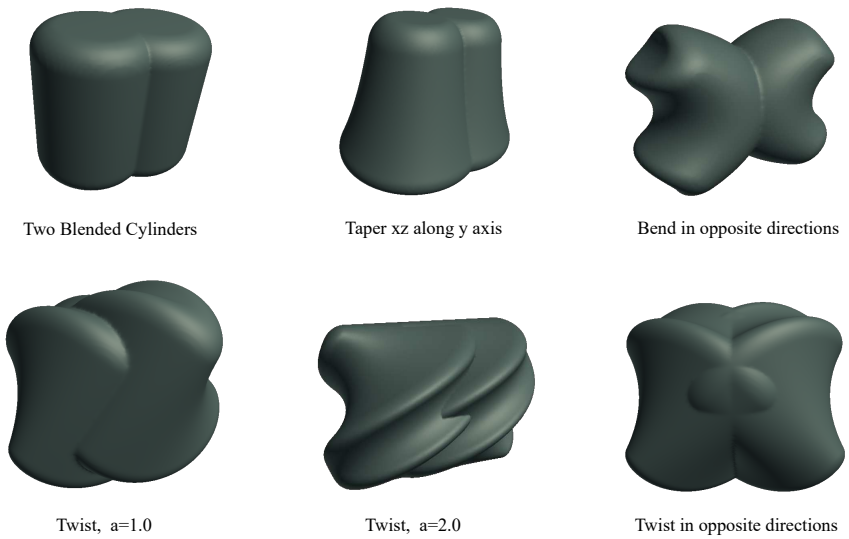


Figure 3.6: Warp operations performed over two blended cylinders.

Chapter 4

Implicit Modeling of Inscribed Volumes

Volumetric modeling of spongy structures refers to a wide range of complex porous constructions of any living or non-living substances which can be morphologically quite different from each other. All Spongy structure materials apart from their morphological, biological, and geographical diversities, contains pores in their bodies which is the main concern of this research. Hence, we skip studying their general appearance, e.g. solid body shapes, color, textures, and similar attributes. There are many porous structures which can be constructed by our approach. Some instances of these supported substances are radiolarian, cancellous bone or spongy bones, volcanic rocks, pumice stone, some types of cheese and other similar structures. The main objective of this research is to apply the 2-D method introduced in [1] to 3-D and by introducing new methods it can be employed in a wider spectrum of applications, such as the examples above.

In this chapter, we describe our techniques to construct many porous or spongy structures where the holes can be defined over the outer surface of a solid object or inside of it; regularly connected or naturally connected with each other. The 2-D implicit method from [1] lacks some generalizations in its definition that can be crucial for a versatile 3-D porous surface construction, meaning that all the cells were assumed isolated from each other which leads to isolated inscribed curves as well. Smooth and convex isolated inscribed volumes (IV) defined within 3-D Voronoi diagram cells in $\Re^3$ to be utilized in a positive or negative form. Positive form refers to the original blobby shapes such as smooth pebbles, and negative form refers to

when they subtracted from another solid primitive to produce sponge-like objects.

We first start with the basic 3-D method which considers each cell isolated from the other cell and defines distance field function for individual cells and later two new methods of creating the interconnection between neighboring cells are introduced. The interconnection between regions gives us a great flexibility to support more varieties of life-like spongy structures. In addition to cell interconnection, it is nice to have sufficient control on the IVs size and smoothness, and to modify the raw field value to make it compatible with other skeletal primitives, particularly when a combination field is required. The improvement techniques for these matters are discussed in Section 4.3.

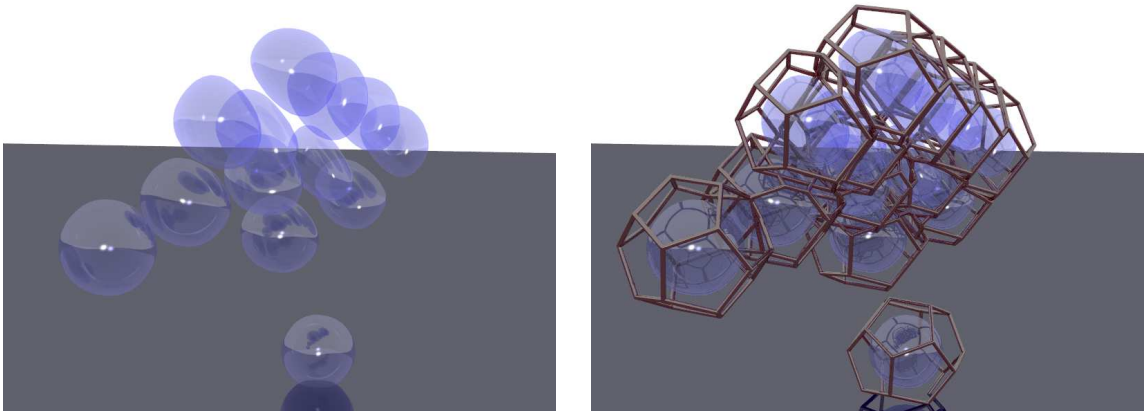
4.1 Isolated Inscribed Volumes

In order to produce our objective of spongy shapes, we start with isolated inscribed volumes and then progress to interconnections. The idea is to have a distance function to generate a set of 2-manifold smooth and closed inscribed volumes where each volume is approximated from the walls of a bounded convex region and must be defined in the interior of the region. The smoothness and roundness of the IVs help to fulfill the aesthetic aspect of the results where having nice and smooth curvatures is a matter of importance which includes an architectural or industrial production of decorative designs; creatures like radiolarian and some living sponges, or objects like pebbles and Swiss cheese. The implicit definition of the function gives versatility and ease of having smooth as well as non-smooth shapes to mimic modeling of the environmental effects, such as weathering or other phenomena on substances found in nature by using surface deformation [11] or any displacement noise functions such as Perlin noise [88].

The mathematical approach of this section is similar to the 2-D implicit method described in the Chapter 3, but the 2-D geometry elements are adjusted to 3-D elements, e.g. bounding polygon to polytope and edges to faces, and we briefly repeat the function definition in 3-D space. The function we define in this section can be improved by adding appropriate adjustments and weighting to produce more accurate shapes as well as field compatibility in field combinations (see Section 4.3). It also worth mentioning that although any convex polytopes could be used, we choose the Voronoi cells as they are easily obtained by a set of points in $\Re^3$. Moreover, the Voronoi data structure can help to derive the neighboring cells connectivity or other

cell identity information with ease comparing with a random set of connected regions.

Given a finite set of point seeds in $\Re^3$, a 3-D Voronoi tessellation is constructed from these points in Euclidean space. In general, Voronoi cells may not be connected to each other or even be convex [14], however, in this research, we consider 3-D Voronoi tessellation based on Euclidean distance measurement between initial seeds which results in convex cells. For the basic function definition (i.e. isolated IVs), the connectivity of the cell regions is inessential, as each region is considered individually. For a more natural modeling of some spongy structures such as cancellous bone tissues or some cheese types, partial or full interconnection between the cells is important. The input points do not directly define the shapes of IVs, yet if they are defined as the seeds of Voronoi tessellation with connected structures of cells, the distributions or patterns of these points and their distances from each other can affect the overall shapes and density of IVs, i.e. volume size, roundness, and regularity vs. stochastic aspect. Figure 4.1 shows a set of dodecahedrons cells. Note that individual cells with the neighbor(s) in their close vicinity have deformed and so their IVs.



(a) (b)

Figure 4.1: A pack of implicit inscribed surfaces within dodecahedrons Voronoi cells. Note that in this example, each cell is individually defined but deforms as the neighbour's seed points get closer.

Assuming a bounded convex polytope $\mathbb{C}$ with finite number of polygons (faces) is provided, f_i is the i'th polygon of $\mathbb{C}$ with the counter clock-wise ordered sequence of its vertices $\mathbf{V}_{f_i} = (\mathbf{v}_0, ...\mathbf{v}_{L_i})$, where L_i is the number of polygon i vertices. If $\mathbf{n}_i$ is the inward unit normal for f_i, then for $\mathbf{x} \in \Re^3$, the signed distance of the face f_i to the point $\mathbf{x}$ is

$$d_i(\mathbf{x}) = (\mathbf{x} - \mathbf{v}_{m_i}) \cdot \mathbf{n}_i, \tag{4.1}$$

where $\mathbf{v}_{m_i}$ is the *m'th* vertex of the *i'th* face of $\mathbb{C}$ and can be chosen from any vertices of the *i'th* face or even the face centroid. For each point located on the face f_i, the distance is zero $d_i(\mathbf{x}) = 0$.

For IV, the function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{C}$, must be positive inside $\mathbb{C}$, zero on the boundaries of $\mathbb{C}$, and unbounded outside of $\mathbb{C}$. Since we need to guarantee the convexity of f, the logarithm function is used to make the function f *logarithmically convex*. The function f can be defined as

$$f(\mathbf{x}) = \log \prod_{0 \le i < k} d_i(\mathbf{x}) = \sum_{0 \le i < k} \log\, d_i(\mathbf{x}), \quad \mathbf{x} \in \mathbb{C}. \tag{4.2}$$

where k is the total number of vertices of $\mathbb{C}$.

The smoothness and size of inscribed iso-surfaces depend on the choice of the value of iso-value a, namely if a approaches to zero then $S_a = \mathbf{x}$, which represents the points on the inscribed volume, provides a closer approximation of polytope $\mathbb{C}$ boundaries and the larger the value of a, the rounder and smoother iso-surface will be produced.

4.2 Interconnected Inscribed Volumes

The aforementioned approach of isolated IVs can go even one step further to break the constraints between connected cells of a tessellation which gives us a remarkable flexibility to construct a wider range of natural substances. For instance, a type of bone tissues in the human skeleton is the Cancellus bone that has a spongy structure to facilitates the movements of the joints, or natural modeling of some types of cheese, volcanic rocks, and similar structures. These complex and random structures are not directly achievable from the function f since the IVs from a tessellated 3-D space are isolated in their own cells, and without the need of having any information from their neighbors. We classify these structures into two categories of *fully-connected* and *partially-connected*.

In this section, two new methods of creating interconnected cells are described where each one is dedicated to constructing one of above categories of a spongy structure. The *fully-connected* method provides a clean, careful, and complete link through the shared faces of all connected cells of a Voronoi diagram by generating extra inscribed primitives. On the other hand, in the *partially-connected* method, the objective is to develop a random and natural link between the IVs which avoids

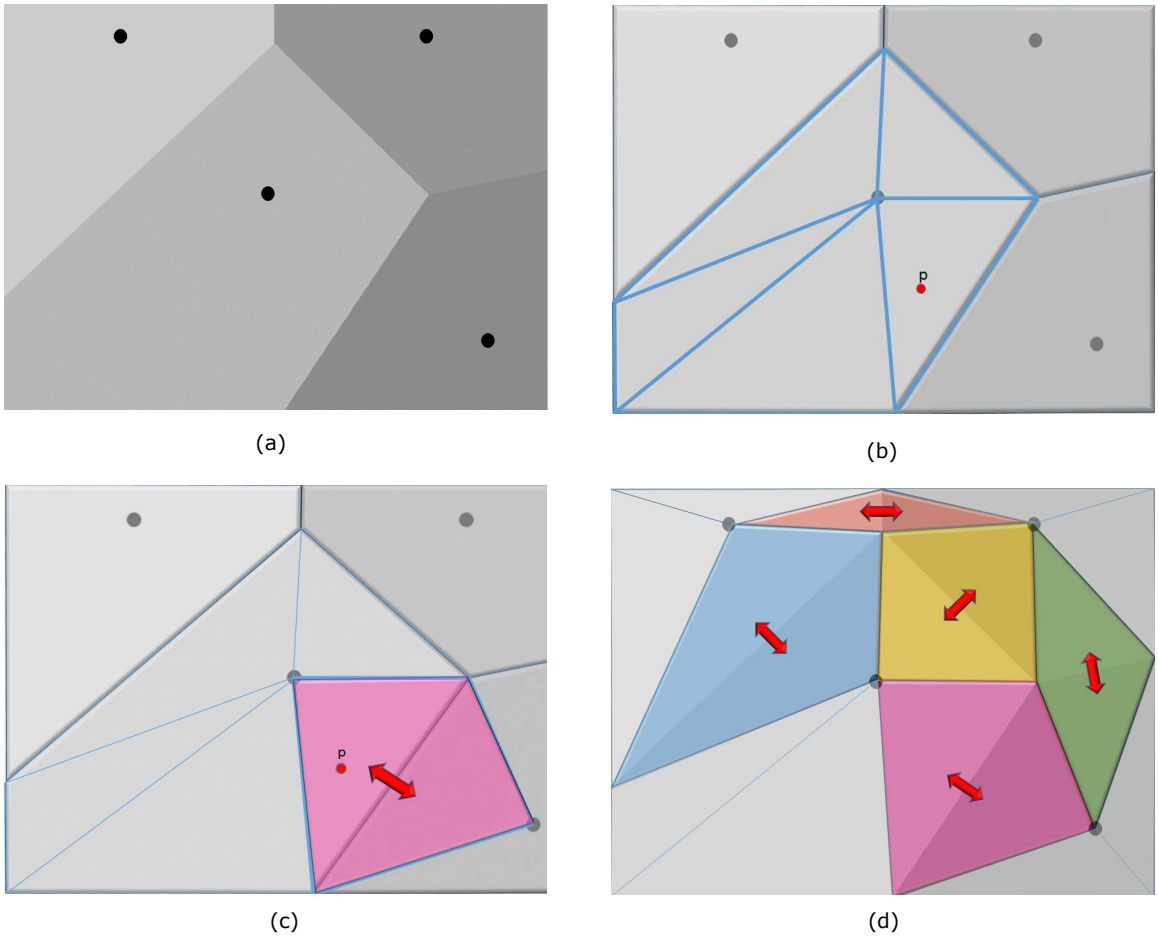


Figure 4.2: 2-D representation of interconnection between neighboring cells. Figure (a) represents four neighboring cells, (b) shows finding the right cage inside cell $\mathbb{C}$ which contains point $\mathbf{p}$, (c) depicts creation of a new cage from the two cell centroids and the shared face shown in pink color, and the figure (d) repeats the steps from (b) and (c) for all neighboring cells in the Voronoi diagram.

connection of all cells. The algorithm of these two methods may appear similar to some extent, yet the resulting shapes represent two distinct types of porous structures.

4.2.1 The Fully-Connected Method

The aim of the *fully-connected* algorithm is to construct a passageway or tunnel between each two neighboring cells through their shared face and produce a network of IVs. Therefore, for an internal cell $\mathbb{C}$ of a 3-D Voronoi tessellation with i faces, there must be i passageways through the faces (one for each face) to the neighboring cells. The passageways fields should smoothly blend with the initial inscribed fields of

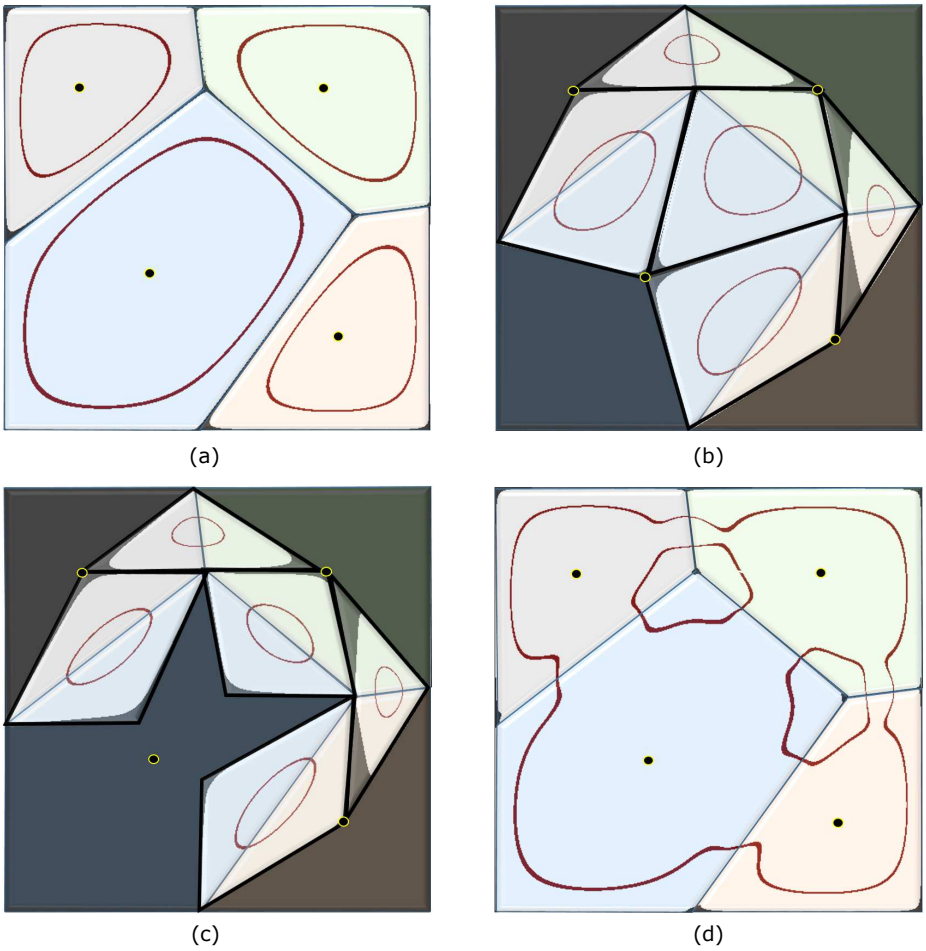


Figure 4.3: The 2-D slice representation of interconnecting IVs using the *fully-connected* technique. Figure (a) represents four initial inscribed curves in dark red color, (b) shows the interconnecting primitives within their corresponding cages from centroid points of the cells, (c) depicts the modification of the cages sizes to provide a balanced contribution of the new primitives in each neighboring cell, and the figure (d) shows the blending of initial curves and the new interconnecting primitives.

$\mathbb{C}$ and have the same field value range to satisfy the field consistency and smoothness of the new IVs. In this method the passageways always pass through the middle of the face and can never break the shared face edges and vertices or pass through other faces, i.e. in the case of two connected cells that are topologically concave as a whole as well as their passageway cage.

To construct such interconnecting primitive, the first step is to divide the cell $\mathbb{C}$ into i cages where i is the number of faces of $\mathbb{C}$. Each cage is constructed from the vertices of the shared face and the centroid point of the $\mathbb{C}$. The next step is to find the right cage cg_i which contains point $\mathbf{p}_{xyz}$ whose field value is queried,

hence we know for which face of $\mathbb{C}$, the new primitive field value must be evaluated. Figure 4.2 (a) and (b) shows this step; note that in this figure the 2-D Voronoi cells are illustrated for simplicity. At this stage, the cg_i should expand in size to enter the corresponding neighbor cell by including the neighbor centroid point to the list of cg_i vertices (Figure 4.2 (c) and (d)). This new cage is a bounded region where another primitive gets inscribed within its walls. The size of this new primitive and its roundness are controlled from the function f in the same way as the initial IV. The blending between these two primitives is controlled by Ricci blending operator (see 3.3.1), which can be adjusted from field summation to the clean union of the two fields.

Figure 4.3 shows the application of the *fully-connected* method on a 2-D slice of the actual 3-D IVs. In this figure, four initial IVs closed by four corresponding Voronoi cells blend with five interconnecting primitives and for the actual blending, in Figure 4.3 (d), the sizes of the tunnel cages are modified in Figure 4.3 (c). This is because the sizes of the some neighboring cells are quite different and for example the bigger cell, in blue color, contains a bigger portion of the interconnecting primitives than its neighboring cells (see Figure 4.3 (b)). To provide a balanced contribution of interconnected primitives for each neighboring cells, the position of one of the two centroid points, which is usually the bigger cell, in the passageway cage can be adjusted toward the shared face centroid point.

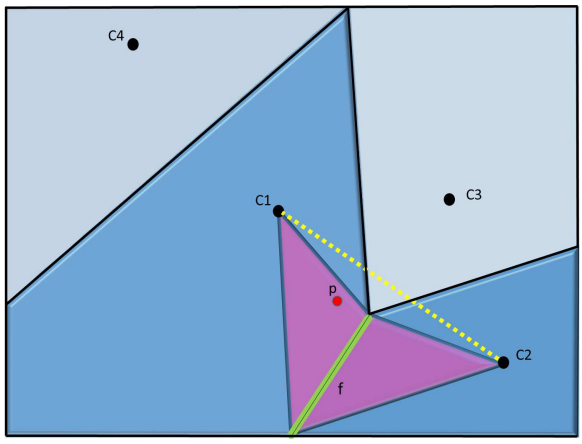


Figure 4.4: The line connecting the centroid points of two convex neighbouring cells (darker blue area) may not pass through the shared face which is the green line in this 2-D representation. In this case, the resulting cage is not convex.

4.2.2 The Partially-Connected Method

In second method, the first two steps are similar to the *fully-connected*, but differs from it after the cage cg_i in $\mathbb{C}$ is found. In this method, two spherical *booster* fields are generated around the two centroid points of the neighboring cells, and they blend with each other to create a new primitive which eventually must blend with the initial IV of the cell $\mathbb{C}$. What distinguishes this method from the previous approach, is a number of variations that can be accomplished to achieve different results with the absence of regularity. This method has a flexibility of modeling randomized and *naturally imperfect* substances. For instance, the holes in the Swiss cheese can be the result of trapped particles of hay during the production process, and therefore their shapes are random and appear as a mixture of isolated and interconnected holes. In such cases, the *fully-connected* method is not an appropriate choice since it lacks such randomness. Note that it is possible to perform a pre-processing on *fully-connected* method to flag some random cells as isolated or partially connected through specific faces, but it may make the method unnecessarily complex as it is not designed for such irregularity.

In the *partially-connected* method, the radius of booster fields is important as they blend with the initial IVs and may produce a large surface that can go beyond the boundaries of the cell and cause unwelcome results (see Figure 4.5 (f)), the same issue also happens when the blending operator is not appropriately chosen. To prevent this issue, the radius of the booster field in each cell must be less than the shortest distance from the centroid to each face or vertices of the same cell. Nonetheless, if this data is not immediately available from heuristic search of the Voronoi data structure and is computationally expensive, the radius must be chosen with sufficient caution or the size of the initial IV is reduced. Note that if the initial (isolated) IV field value reduces dramatically, it makes the blended surface spherically shaped which depending on the usage, may not be desired. The combination of having two cell centroids that are far from each other, and restricting the sizes of the booster fields, the two neighboring cells may not connect with each other which helps with having a more random and partially connected network of IVs. An optional adjustment is to prevent blending of two connecting cells where the line between two centroids does not pass through their shared face (see Figure 4.4). This step involves extra line-polygon intersection calculations and can be optionally executed.

A faster and less regular variation of this method is to avoid the use of the cage

concept and to blend the booster field value of cell $\mathbb{C}$ with its all neighbors' booster fields and finally blending the result with the initial inscribed field. For this variation, it would be helpful to have an accurate blending operator such as *gradient-based blending* by Gourmel et al. [4] to avoid unwanted bulging artifacts of most blending operators. A simple and less accurate alternative is a carefully chosen size of the booster fields and the initial IV field. This variation works best for natural modeling of chaotic porous structures such as weathered volcanic rocks.

Figure 4.5 shows an example of *partially-connected* with the same initial inscribed curve (volumes) as in the Figure 4.3 (a). This figures represents the interconnection between neighboring cells using the cage concept (4.5 (a) and (b)), without the cage concept with the right sizes of interconnecting or booster primitives (4.5 (c) and (d)) and when the size of booster primitive are so big, so the blended surface goes beyond the boundaries of the cell(s) (4.5 (e) and (f)).

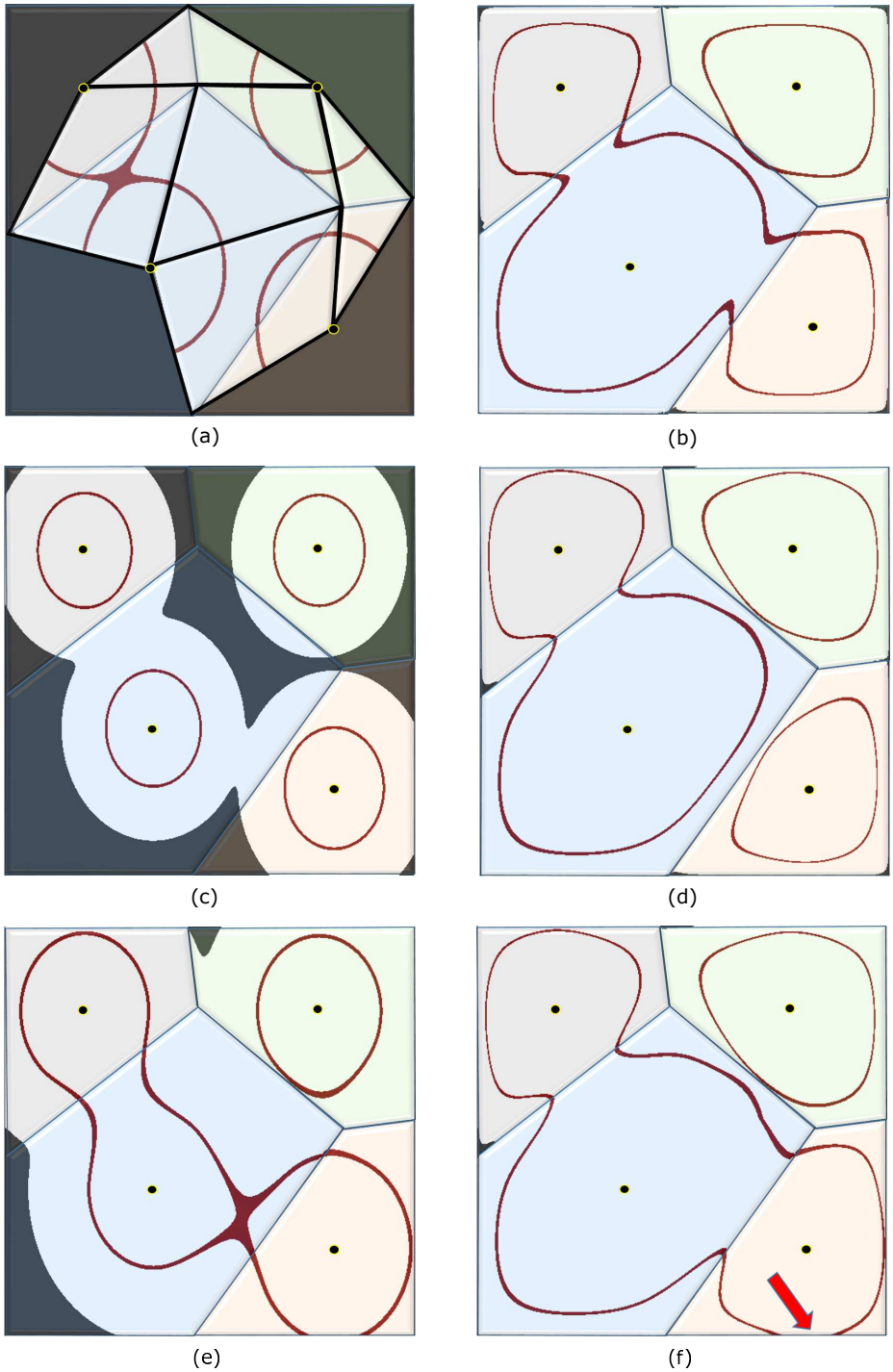


Figure 4.5: The 2-D slice representation of interconnecting IVs using *partially-connected* technique. The right column shapes shows blending of left column shapes with Figure 4.3 (a) initial IVs. Figures (a) and (b) represent using cages as in *fully-connected* for booster fields and after blending, (c) and (d) depict usage of booster fields for interconnection without the cage concept, and the figures (e) and (f) show the wrong choice of the size of one booster field, in the light orange cell, and passing beyond the cell walls after blending (marked by the red arrow).

4.3 Improvement Details

Practically speaking, it is necessary that the function (4.2) provides a precise evaluation of its bounded region and gives the flexibility to control the size and smoothness and be compatible with other skeletal primitives. A careful look at (4.2) reveals that:

1. when $\mathbb{C}$ contains one or more tiny faces comparing the average face sizes of $\mathbb{C}$, it is visually expected to have a coefficient to force the tiny faces to have the smaller effect on the shape of IV and vice-versa for larger faces. This issue can be handled the same way as the 2-D method by adding an edge, which in our case it would be face weight to the (4.2), however, we used some new weight coefficients to have an even better control of the IVs, such as size and smoothness controls. Previously, the iso-value a was responsible for controlling the sizes of IVs, but using the locally supported functions with a fixed iso-value of 0.5, requires us to define a function coefficient to facilitate the IV controllability;

2. using the logarithm function may produce big negative field values for distances less than 1, and positive big values which make the field continuity less smooth and less compatible with other primitives in the BlobTree that have local supported field ranges. The reason for making the field restricted is that the IVs, by definition, are quite different from skeletal primitives. The field value for skeletal primitives are defined on the outer region of the skeleton geometry with a fixed radius, while the inscribed volume, as the name suggests, are defined in the interior region of a 3-D geometry and there is not a fixed radius, i.e. a fixed distance from the bounded region may also affect the smoothness of the shapes. Note that the logarithm function can be replaced by any other function in $\Re^3$ that fulfills our criteria, which is being convex and bounded in interior region of $\mathbb{C}$, and zero or unbounded over the boundaries of $\mathbb{C}$. In this section, we only focus on logarithmic field range adjustments.

The field value of f is derived from its distance to each face of $\mathbb{C}$, but what is missing from this function is the importance of each face in the overall resulting surface. Given polytope $\mathbb{C}_A$ and $\mathbb{C}_B$ with the exact geometry; If in the cell $\mathbb{C}_B$ the area size of face f_i gets a bit smaller, or a tiny face gets added to the list of its faces, the generated iso-surface will be modified as if a big change has occurred to $\mathbb{C}_B$ comparing with $\mathbb{C}_A$. This is simply because all faces of $\mathbb{C}_B$ have the same influence

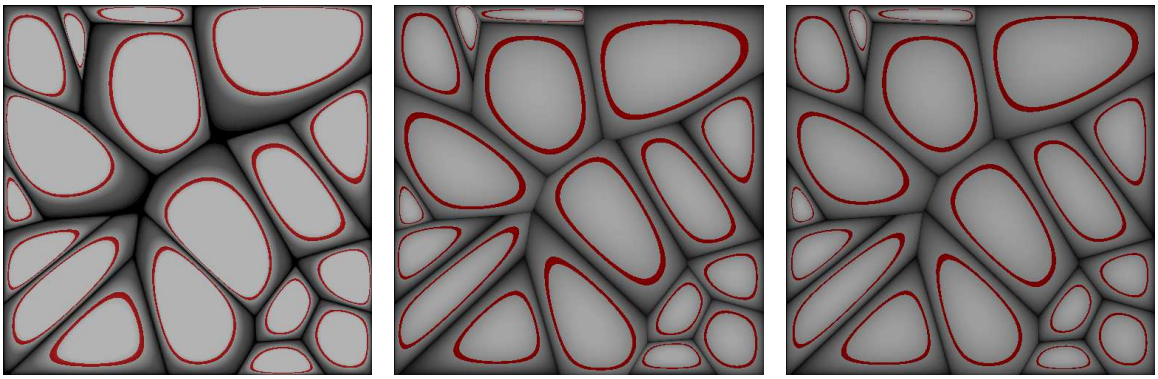
on the function f, therefore no matter how insignificant effect a small face has on overall geometry of the $\mathbb{C}_B$, the generated surface represents a big impact of that face on $\mathbb{C}_B$. The simple solution to this issue is to consider a coefficient for each face, so the inscribed iso-surface changes according to the weight of each face. Consider

$$f(x) = \sum_{0 \leq i < k} \lambda_i \log d_i(\mathbf{x}), \tag{4.3}$$

where λ_i indicates the *i-th* face area over the total cell area ($\lambda_i = A_{f_i}/A_{\mathbb{C}}$). According to (4.3), the faces with bigger area have more affect to the inscribed surface than the smaller faces. To gain a better control of influence of face areas, λ_i can be modified to

$$\lambda_i = A^c_{(f_i)}/A_{\mathbb{C}}, \quad 0 < c \leq 1. \tag{4.4}$$

The value of c can affect the overall volume size of inscribed surfaces as well, so a careful choice of c can provide a smooth and well approximation of $\mathbb{C}$. In practice, we found $c = 0.7$ combined with other adjusting factors, such as iso-value and another shape control weighting, which will be explained shortly in this section, can fulfill those criteria (Figure 4.6).



(a) (b) (c)

Figure 4.6: A 2-D slice of iso-surface with 0.5 value. Figure (a) represents the function f prior applying face area ratios, (b) after considering face area ratios, (c) the function f using each face ratio with a power value of 0.7, pushing the contour towards the center space.

Another factor that was overlooked in the function (4.2) is the value of $d_i(\mathbf{x})$, which comes from the distance of point $\mathbf{x} \in \mathbb{C}$ to the face f_i and is calculated from the *dot product* of $\mathbf{n}_i$ and $\mathbf{x}$. A negative value will be discarded since it represents a

point $\mathbf{x}$ outside of $\mathbb{C}$, but large positive values can produce large and unwanted values of function f. To prevent having too large $d_i(\mathbf{x})$ value, the centroid $\mathbf{c}_{\mathbb{C}}$ of $\mathbb{C}$ can be taken into account and $d_i(\mathbf{x})$ is redefined as

$$d_i(\mathbf{x}) = (\mathbf{x} - \mathbf{v}_{m_i}) \cdot \mathbf{n}_i / (\mathbf{c}_{\mathbb{C}} - \mathbf{v}_{m_i}) \cdot \mathbf{n}_i, \tag{4.5}$$

where the $\mathbf{v}_{m_i}$ similar to (4.1) is the *m'th* vertex of the *i'th* face of $\mathbb{C}$.

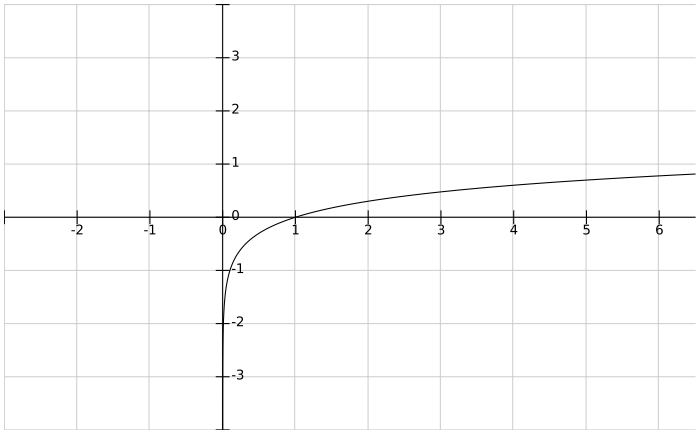


Figure 4.7: Logarithm Function.

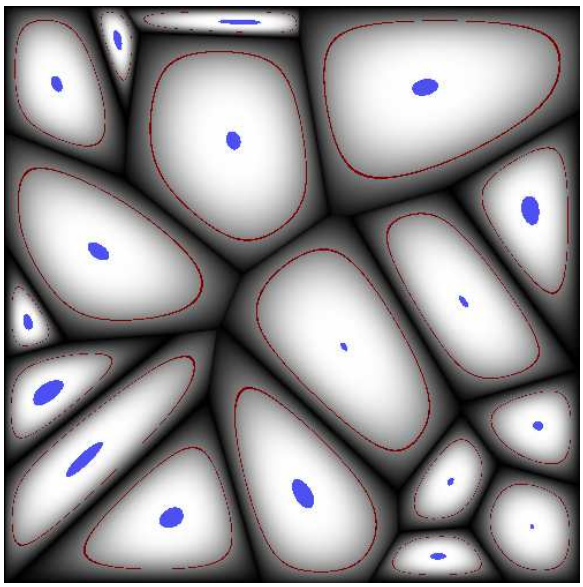


Figure 4.8: In the small regions (blue color) around centroid points of the cells, the field value can be greater than 1.

In this way, the $d_i(\mathbf{x})$ cannot produce big values, and on the centroid $c_{\mathbb{C}}$ it is always equal to 1.0. Furthermore, by looking at the logarithm function, we know that the

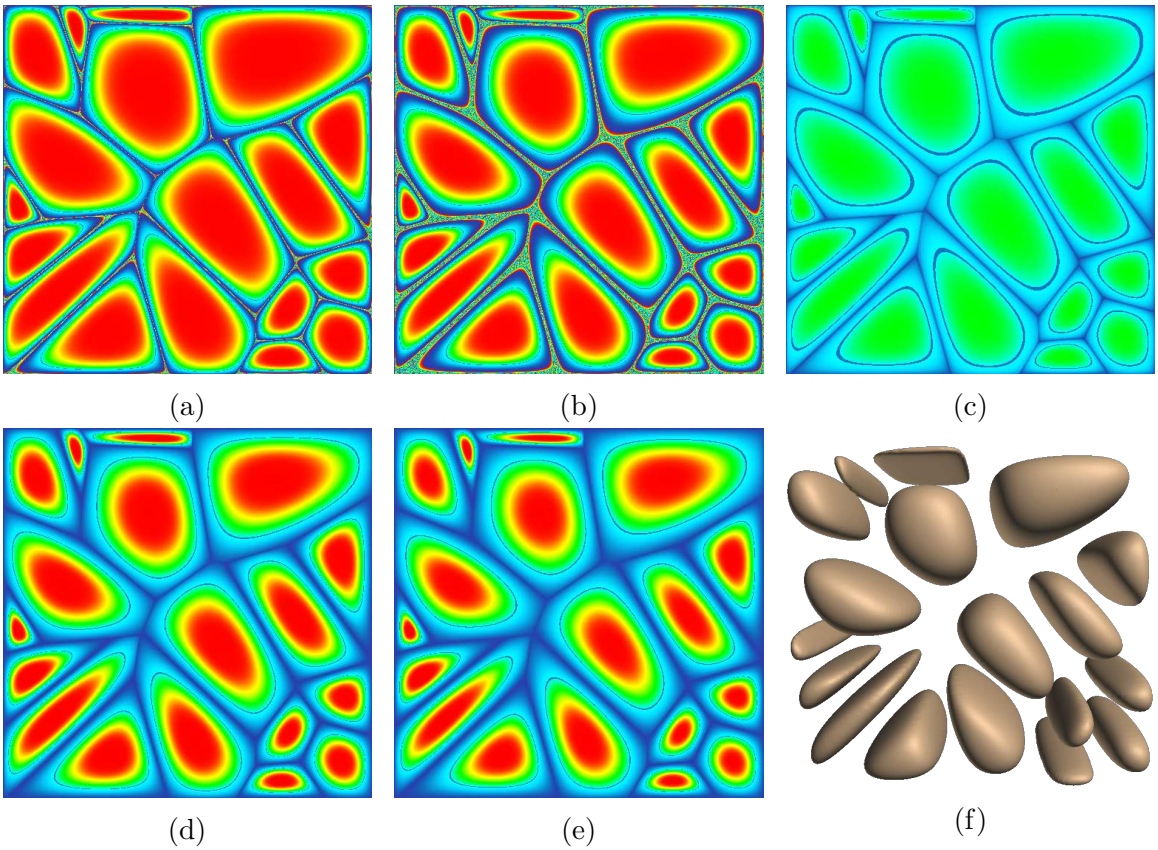


Figure 4.9: A 2-D slice visualization of field distributions. The warmer colors are used for representing the higher field values and vice-versa: (a) function f using log(x), (b) using log($\mathbf{x}$) function similar to figure (a) after multiplying by a function weight (i.e. $0 < \gamma < 1$), (c) using log($\mathbf{x}$+1) while having a small range of $\mathbf{x}$ makes the field values to be very close to each other, (d) using function (4.8) and Wyvill fall-off filter function [5] applied over unsigned distance fields, (e) using (4.8) with Soft Objects fall-off filter function, (f) a 3-D representation of figure (e).

log($\mathbf{x}$) for $\mathbf{x} > 1$ is positive and grows slowly, but for $0 < \mathbf{x} < 1$ it gets arbitrarily close to large negative values and grows very fast. To have a better transition of output values of log($\mathbf{x}$), consider log($\mathbf{x} + 1$) to achieve only positive values. The function f can be changed to

$$f(\mathbf{x}) = \beta \log \Big(\prod_{0 \le i < k} d_i(\mathbf{x})^{\lambda_i} + 1 \Big), \quad 0 < \beta \le 1. \tag{4.6}$$

Nevertheless, using the function (4.6) and having a narrow range of $d_i(\mathbf{x})$, the log function is not capable of producing well field values distributions (see Figure 4.9(c)). An alternative could be considered as

$$t(\mathbf{x}) = \prod_{0 \le i < k} d_i(\mathbf{x})^{\lambda_i}. \tag{4.7}$$

The value of $t(\mathbf{x})$ from (4.7) is always positive, but in our experiments we noticed that in a small region around the centroid points of the cells, the field value can be greater than 1 and less than 1.5, therefore it is clamped to 1 to have a fixed field range of [0,1] (see Figure 4.8). Using log function for $t(\mathbf{x})$, the function f can be defined as

$$f(\mathbf{x}) = -\log\left(b - t(\mathbf{x})\right) / \log\left(b - 1.0\right), \quad 1 < b < 2, \tag{4.8}$$

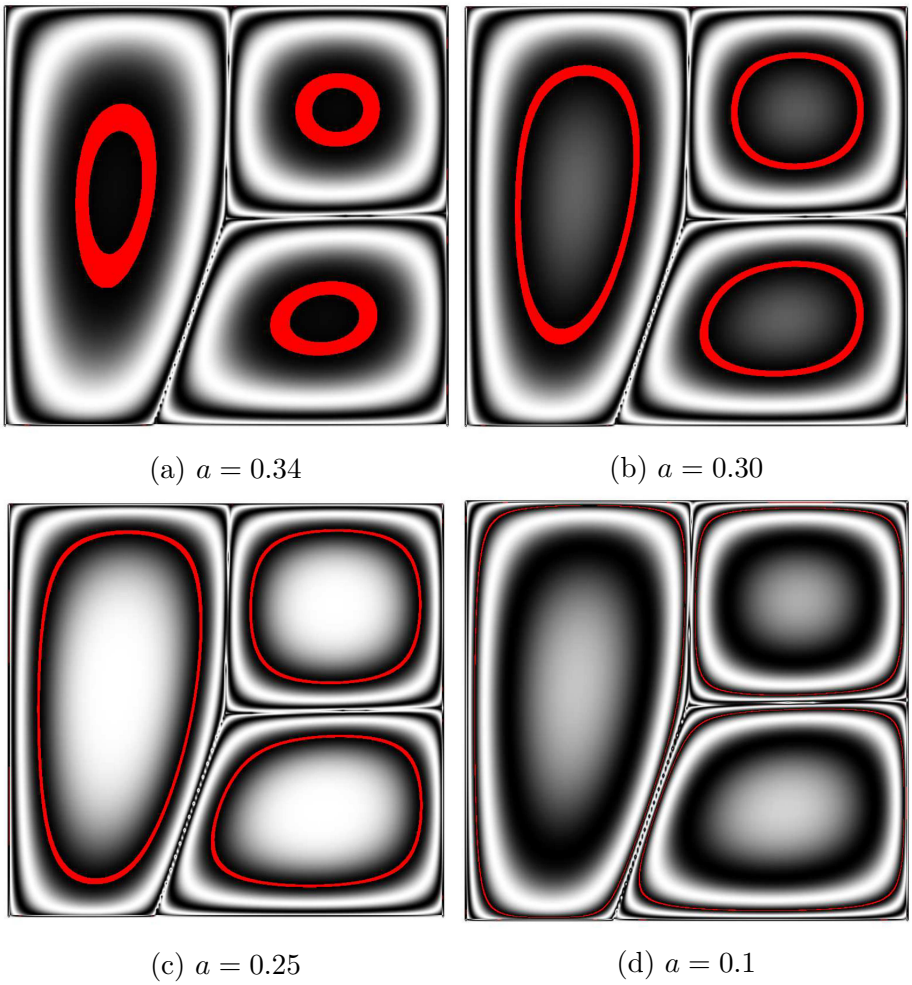
where b is used to control the smoothness and size of iso-surface and combination of it with c value from (4.4), a smooth and appealing iso-surface can be generated. Figure 4.9 plots 2-D level-set graphs from a middle cut slice of 3-D iso-surfaces, it compares the field values of different variations of function $f(\mathbf{x})$, the bottom figures illustrate function (4.8) using Wyvill [5] and Soft Objects [34] fall-off filter function.

4.4 Results and Variations

The techniques described in this chapter allow the freedom to control the shapes size, smoothness, and blending operator to customize the IVs according to the needs of a user. The first decision to make is determining the distribution and density pattern of IVs and, e.g. uniform patterns for synthetic foams or natural honeycombs and random distributions for many natural porous structures. The closeness of input Voronoi diagram seeds produce denser porous materials and their orientations can be reflected in the orientations of IVs. Choosing the right point distribution algorithms varies from one usage to another and our approach can be used regardless of the input pattern.

The size and smoothness of IVs can be adjusted by changing the iso-values (see Figure 4.10), or by changing the weights of function f as explained in the previous section. In general, as the IVs approaches the boundaries their shapes appear less smooth and round and therefore less aesthetic. This may not be an issue for some applications, however, if the aesthetic aspect of the results is the matter of importance, i.e. in furniture or architectural designs, then choosing the best function weights would be necessary.

As it was mentioned earlier, the IVs in their natural form (positive) represent



(a) $a = 0.34$ (b) $a = 0.30$

(c) $a = 0.25$ (d) $a = 0.1$

Figure 4.10: The inscribed iso-contours within three Voronoi cells: as a approaches zero, the S_a which is defined by the red iso-contour, reflects a better approximation of the cells boundary however it may not be a nice and smooth one.

pebble-like objects, but when they get subtracted from another solid object (negative or inverted form), they mimic different types of porous structures. Figure 4.11 represents 200 IVs (left), and one single blob (right) within 3-D Voronoi convex cell(s) in their normal forms, and Figure 4.12 shows porous triple torus shells as negative IVs. In this figure a smaller sized torus is subtracted from a bigger one with the same origin and thickness ratio to produce a thin shell, then a set of inscribed blobs are subtracted from each shell. As it can be seen from this figure, the edges around the holes are smooth and softly graduated toward the center as a result of using the negative blending of two surfaces instead of standard CSG operations, while in Figure 4.13 a standard CSG difference operation was used to create sharp edges in a cheese.

The amount of smoothness in a difference operator can also be controlled by choosing appropriate parameter(s). Note that almost all objects in this work are defined by locally supported field functions with $a = 0.5$ iso-value, and used either the Wyvill fall-off filter function [5] or Soft-Objects field function. The combined surfaces have used BlobTree to exploit its hierarchical data structure.

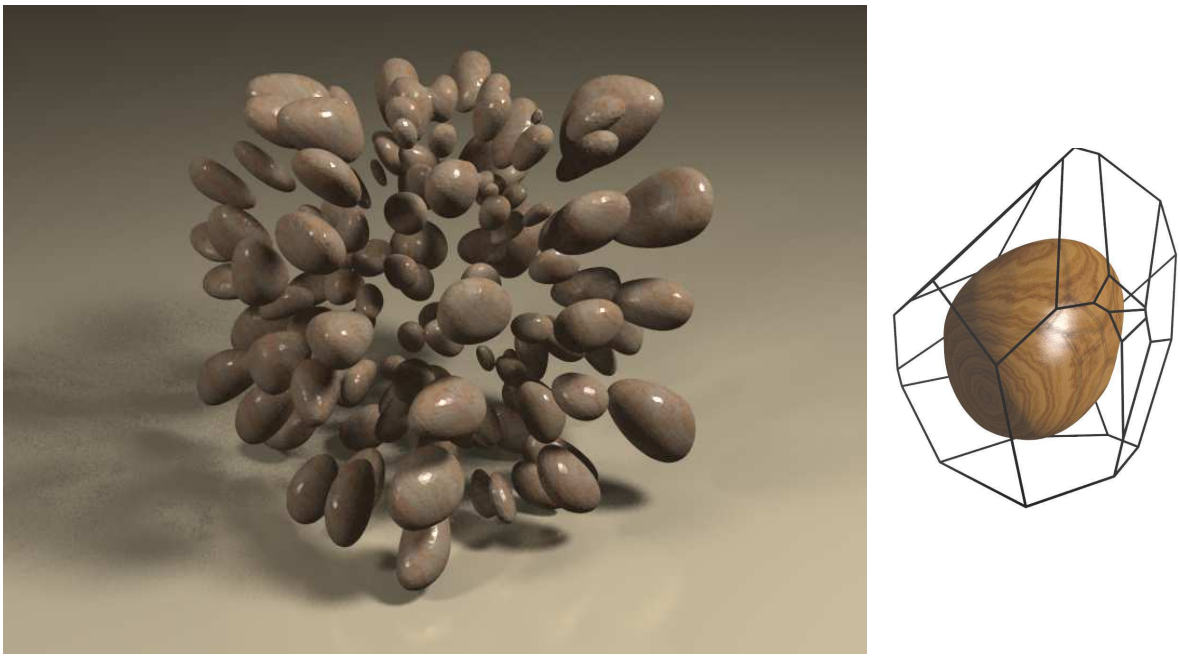


Figure 4.11: Blobby implicit inscribed volumes in positive forms within 200 convex cells (left) and one convex custom cell (right) of a 3-D Voronoi diagram.

The blending operator not only affects the smoothness of the inverted pores in a spongy structure, but it has a crucial role when they blend with each other using one of the two previously described interconnection techniques. As mentioned earlier we used the Ricci blending operator (3.10) which is a straightforward way to control the amount of blending. Figure 4.14 represents the 2-D slice of IVs field values with different blending parameters and caged inscribed primitive sizes using *fully (regular)- connected* method. Note that in this figure three random values of blending parameters and sizes are chosen, while various combinations of blending and inscribed primitive sizes are possible. Figure 4.15 shows a 3-D version of these four IVs in their Voronoi cells blends with each other using caged primitives. In this figure, the size of initial IVs shown in Figure 4.15 (a) became bigger in Figure 4.15 (e) to illustrate how changing the size of initial primitives can affect the overall shape. Figure 4.16 depicts the blended IVs in positive and negative forms.

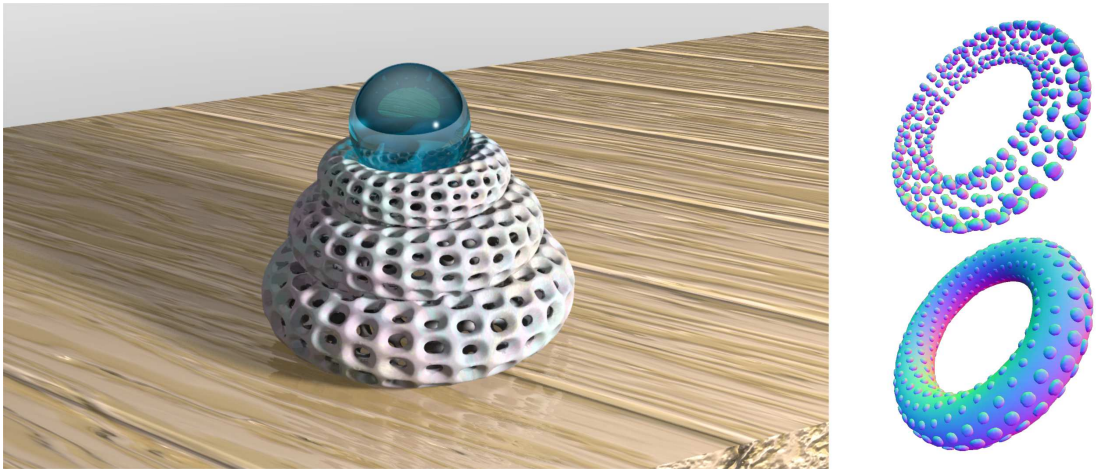


Figure 4.12: Porous triple torus shells with smooth pores.

Figure 4.13: A cheese with isolated and sharp edge holes.

The variations in *partially (semi)-connected* method depends on the power (size) of neighboring booster fields before and after blending, and the blending parameter

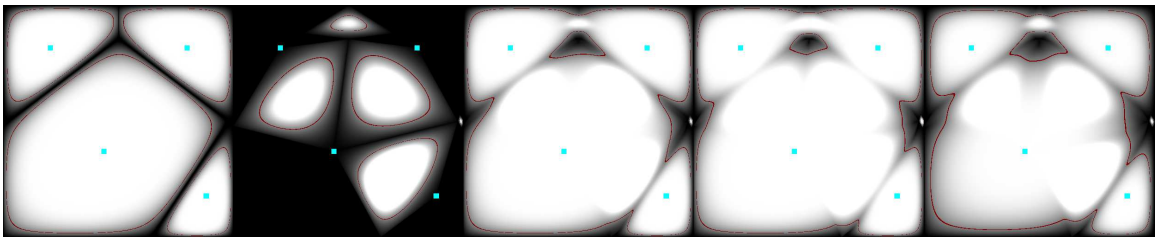


Figure 4.14: 2-D slice of *Regular connected* technique field values from left to right: the initial IVs , new inscribed primitives within bounded cages, blended initial and caged primitives, using bigger size of caged primitives and smaller blending parameter in Ricci blending, and smaller size of caged primitives with a bit smaller blending parameter.

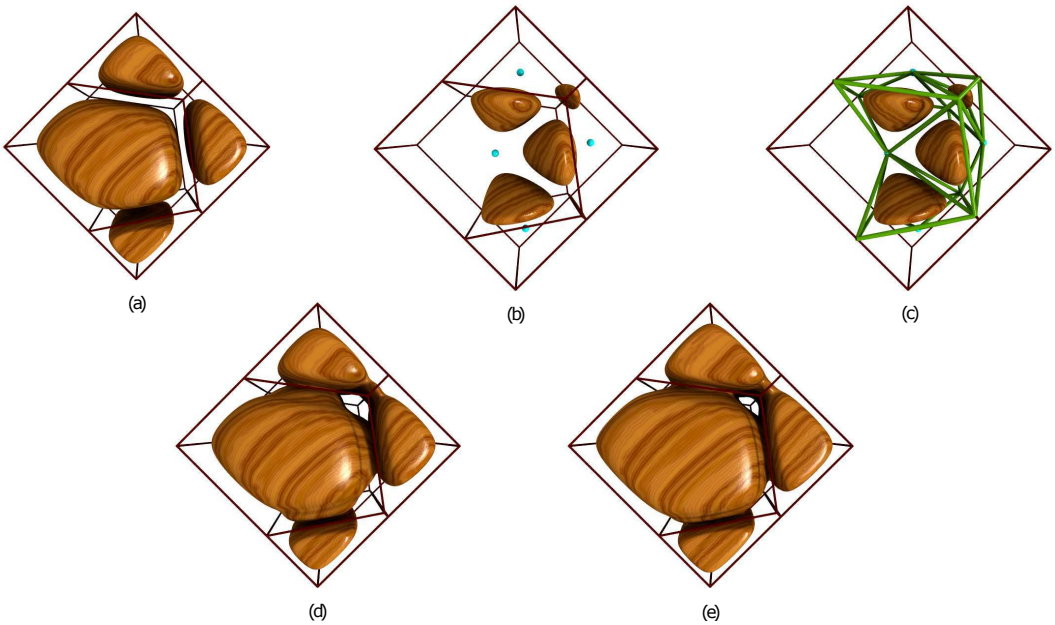


Figure 4.15: *Regular connected* technique for blending neighboring IVs: the initial IVs (a), new inscribed primitives within bounded green cages (b) and (c), a network of blended initial and new primitives (d), blended shape with bigger initial IVS (e).

that is used between them and the initial IVs. In addition, whether the cage concept is used for this method or not can affect the blended shapes. The top row images in Figure 4.17 shows using this technique with and without connecting cages. Apart from blending and shape sizing settings for this method, it is also possible to check the field value of the middle point on the line between two centroid points and if this amount is not greater than the iso-value (i.e. they are not blended at all), we can prevent blending them together and also blend with their corresponding initial IVs, however, in the Figure 4.17 this step is dismissed to show that they could still get blended with

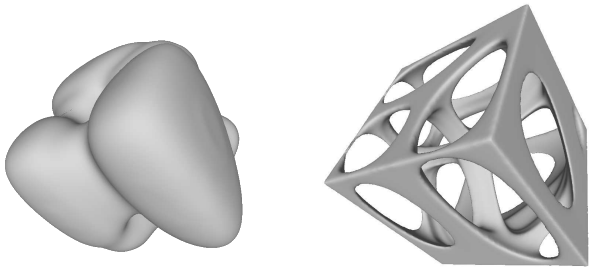


Figure 4.16: *Regular connected* technique for blending four points: positive form (left), negative or inverted form (right).

the initial IVs. Since this approach is used for natural porous materials such as rocks or cheese (see Chapter 5), the regularity of blended shapes is not mandatory which gives the user a flexibility to control or ignore different these types of parameters. Figure 4.18 shows the IVs in negative (inverted) forms and inside of the object in positive forms.

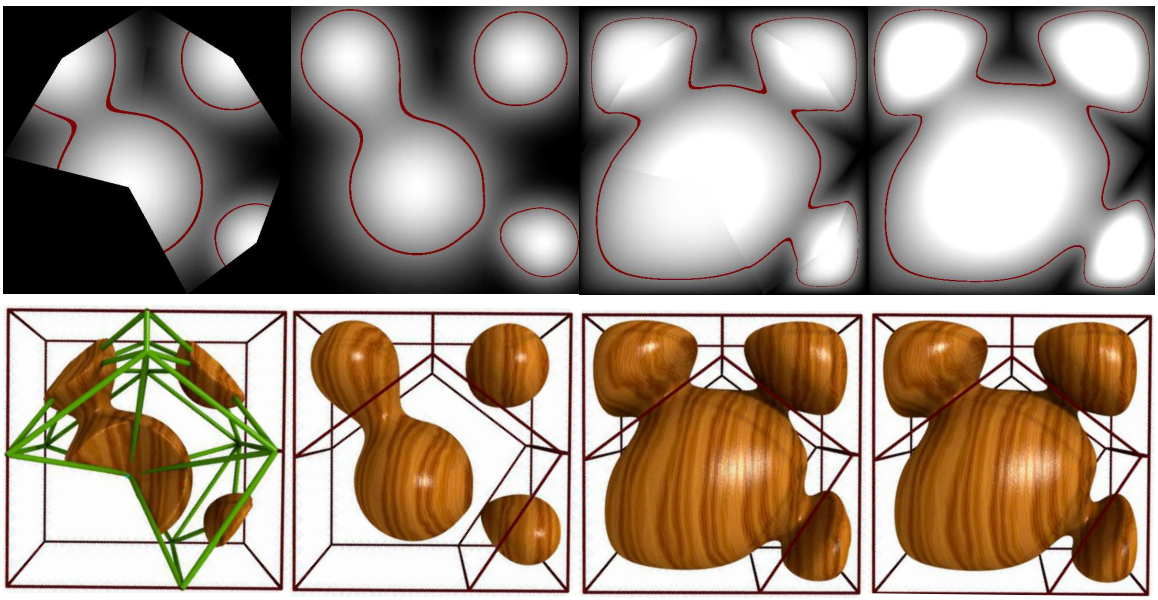


Figure 4.17: *Partially-connected* technique for blending neighboring IVs. The top row images show the 2-D slice of 3-D shapes in bottom row. From left to right: the booster fields within the shared face cages, the booster fields without considering the cages, blended shapes using caged connecting fields (less consistent), blended shapes by ignoring the connecting cages.

Controlling the blending between initial IVs and the connecting primitives can

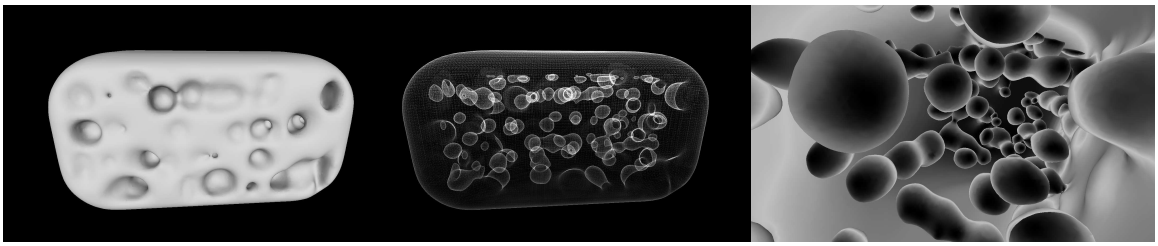


Figure 4.18: *Partially-connected* technique: blended IVs in a negative form, X-ray like image of all pores, camera inside of the object to show interior IVs.

only affect those shapes that are big enough to be blended with each other in both interconnecting techniques. Different settings can influence the quality of blending or prevent it from happening, such as too small connecting or IVs primitives, using blending parameters that are very close to clean-union operator, too far centroid points in *partially-connected* method, very small shared face or cage in *fully-connected* method, and dramatic sizing difference between neighboring IVs (similar to Figure 4.15). Thus, it is important to choose appropriate settings or Voronoi seed distributions to achieve desired effects as some of the mentioned factors can be required for some practices and unwelcome in some others. In Chapter 5, more examples of our approaches are presented.

4.5 Summary

In this chapter, we proposed a versatile volumetric modeling method to construct smooth inscribed surfaces within bounded convex cells of Voronoi diagram followed by two generalization techniques to demonstrate its flexibility in various applications. The IVs can be in either form of positive or negative, where positive form refers to smooth blobby volumes suitable for modeling natural pebbles or any similar objects and negative form refers to a solid object hollowed by IVs to construct various spongy structures. Presenting a versatile 3-D modeling of porous structures is a difficult task, as these structures, especially in nature, can be very complex and distinct. Our approach offers a sufficient level of freedom to control the size and smoothness of IVs and to the intuitive modeling of regular, semi-chaotic, and chaotic porous structures. Moreover, using Voronoi diagram not only helps with having geometrically naturally shaped IVs, but the topological distribution of its seeds assists with customized or random topological patterns of IVs as well.

The mathematical definition of Implicit inscribed volumes offers remarkable flexibilities and accuracy when it comes to surface alteration, to customize the modeling construction process. This is particularly important when a natural and irregular surface displacement or a seamless surface combination is required. To guarantee that the resulting shapes have smooth curvature and are C^2 continuous, an appropriate fall-off filter function (Wyvill function) is performed. The BlobTree hierarchical modeling has enabled us to achieve the desired shape composed of IVs, skeletal primitive, and unary or binary operations in an effective and efficient manner.

Chapter 5

Results and Discussions

The results and discussions in this chapter evaluate the capabilities of inscribed volumes approach, which was explained in the Chapter 4, to be used for different porous substances. In general, a versatile approach for modeling porous structures should be able to produce most common types of porous materials and provides some proper adjustment controls for the user to design desired shapes. We briefly compare some of the most related techniques for representing these structures, as mentioned in Chapter 2, with our method to assess the limitations and capabilities of each approach.

5.1 Results

Apart from user input for the distributions and sizes of the IVs, some other factors should be taken into consideration. Whether a porous structure has pores on its outer shell or inside the body; or taking a decision on the morphology of the pore connections and their smoothness controls are some examples.

In a bounded region in 3-D space, the IVs can be randomly distributed in this region, as most examples in this document, which means that the resulting sponge, i.e. IVs in negative form, can have pores in the interior of an object as well. As in the Chapter 4, we divided our proposed method into isolated and interconnected IVs, in this section the resulting figures of each method is separated as well.

5.1.1 Isolated Approach

Figure 5.1 shows when a slice of a sphere is taken from it and the interior holes are disclosed and Figure 5.2 is an x-ray like illustration of the same object in two different

angels to represent all the interior and exterior pores. This characteristic makes our method differ from some methods based on 2-D [19, 22], or 3-D texture synthesis [23–25], where the pores are located on the exterior part of a model. In general, the key which makes these texture mapping techniques distinct from each other is the ability to control different parameters to generate realistic looking porous structures. A recent work using 3-D textures proposed by Bravalle et al. [25] generates 3-D volumetric textures for porous materials. The authors describe how their approach has the flexibility to control sizing, orientation, separation and distribution of the input particles. Their method produces natural results for modeling bread, stones, and similar materials, however, the generated 3-D textures does not represent round and smooth pores of some spongy surfaces such as cheese or internal structures of spongy bones or similar structures. The advantages of parametrization of textures for porous materials is the ability to represent microstructure details over a surface effectively since it can produce natural results as the focus is on the variability of porosity for a 2-D slice of an object. Similar parameters can be employed in our method to generate quality patterns as well.

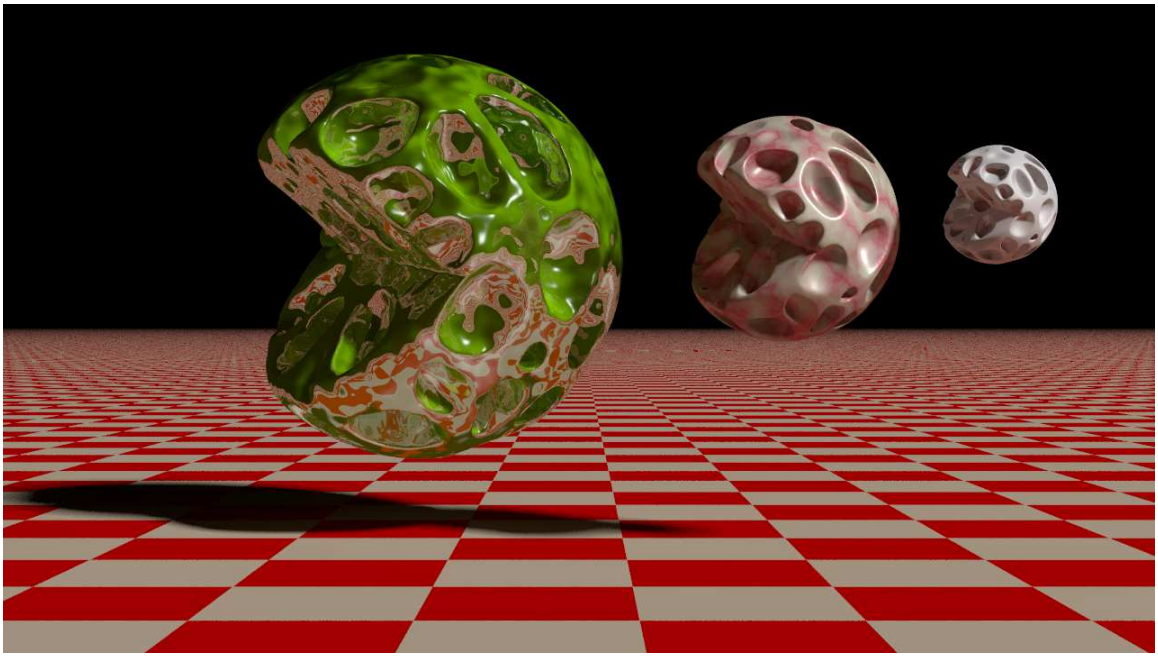


Figure 5.1: Pac-Man: a slice is taken from a porous sphere showing the interior holes.

While the IVs can be defined inside of a model, there is no rule to prevent the user to scatter them uniformly, randomly, or customizing them over the exterior of a surface. In this way, we can reduce the computational costs to evaluate interior

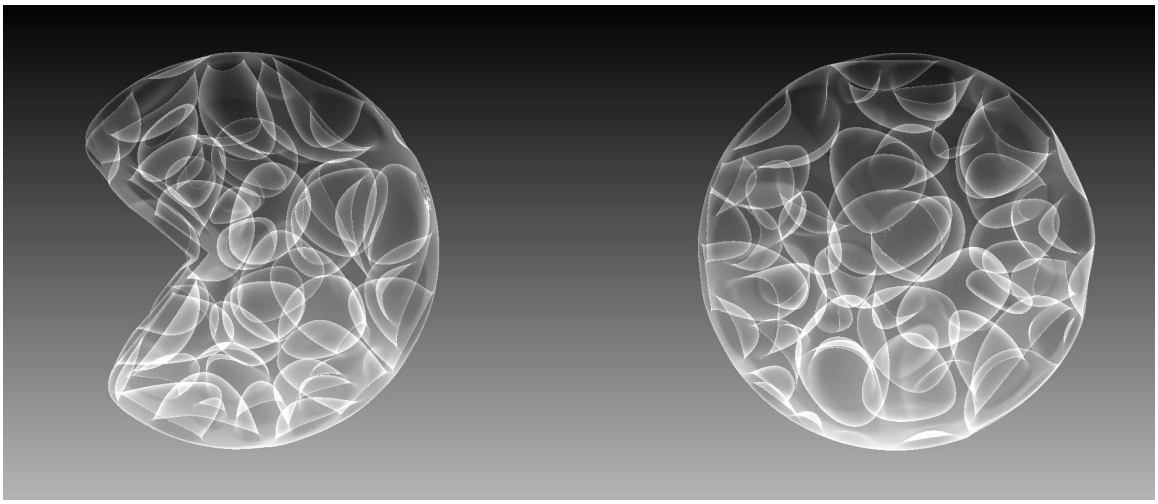


Figure 5.2: X-Ray like images of interior and exterior inscribed volumes in side and back angles.

volumes and increase the amount of control points for IVS where it is required. The Figure 5.3 shows 400 individual IVs distributed uniformly and with specific orientation over a torus surface. As it can be seen in this figure, a uniform distribution of the Voronoi control points produces almost the same sized holes which make this method convincingly an easy choice for modeling natural honeycombs and radiolarians. Figure 5.4 shows a simple model inspired by a type of *radiolarians* called *polycystines*, the fossil of these marine planktons can be found in most oceans and some other shallow waters [89]. They usually have hollowed skeletons made of opaline silica, spines on the outer shells, and a central capsule which separates their inner and outer portions.

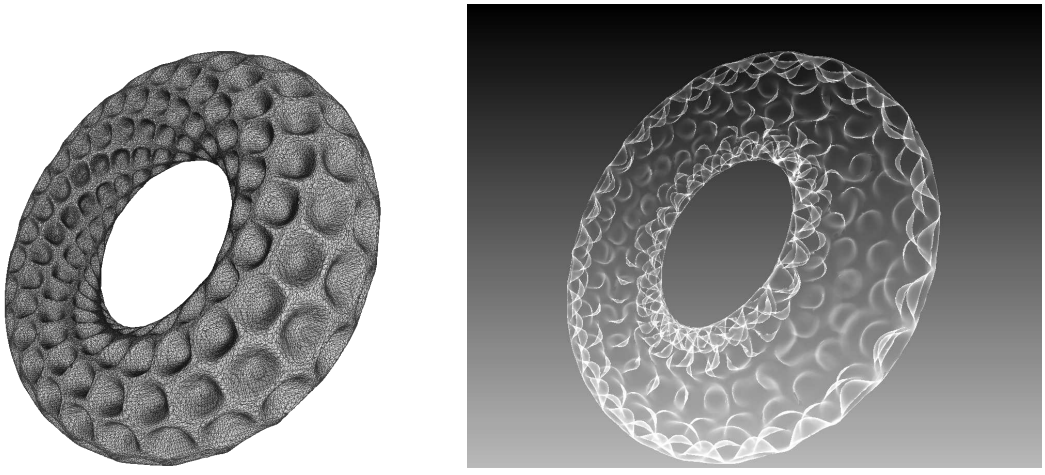


Figure 5.3: Oriented pores over the outer layer of a torus.

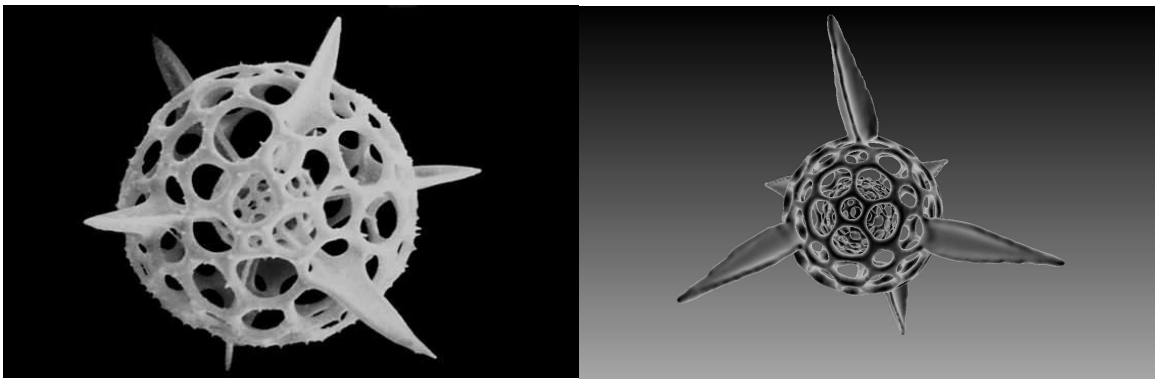


Figure 5.4: A model inspired by a type of radiolarians called polycystines (right) with its main spines and round pores on the outer and central shells; real-life radiolarian (left).

5.1.2 Interconnected Approaches

For those structures whose pores are not completely smooth or round-ish, or have short tunnels connecting pore chambers, the regular or partial connection techniques described in Chapter 4 are used. These algorithms are simple and yet flexible to construct natural looking sponges such as dry foams, spongy bones, some aquatic sponges such as *Clionaidae*, some kinds of cheese, volcanic rocks or pumice stones.

Partially-Connected Results

Figure 5.5 shows a piece of cheese constructed from a half smooth cylinder and cut from top and one side. The IVs are naturally connected with each other using partial connecting algorithm; the sizes of holes and the amount of blending power between the booster fields inside each cell and the IVS, are chosen based on our preferences and are adjustable. Figure shows 5.6 two porous rocks, both created from the partial connecting algorithm which can produce a chaotic representation of pores. The left rock in this figure is a disfigured sphere by 3-D Perlin noise [88], where the value of noise is clamped between [0,1] and then combined with the original sphere distance field. Due to the implicit definition, other methods of surface displacement can be employed to gain natural effect for rocks or other surfaces.

Fully-Connected Results

Figure 5.7 depicts a cross-section of a normal Cancellous bone, which is one of the two types of bone in human skeleton, model inside a compact or *Cortical bone*. For this

Figure 5.5: A piece of cheese with natural partial holes interconnections rendered with subsurface scattering (SSS) technique (right); a real-life cheese (left).

Figure 5.6: Two porous rocks, the left rock is a sphere disfigured with 3-D Perlin noise and the right rock is a sheared sphere (right); four real-life rocks (left).

model, a regular or *fully-connected* method is considered, as the bone structure has smoothed chambers connected with each other through some short tunnels and the same method can be used for Clionaidae sponges or *Entobia* fossils but with tighter tunnels. The Voronoi open-cells based methods for producing dry foams [28–31] are the closest approaches for volumetric modeling of these structures. As it is discussed in Chapter 2, these methods cannot produce smooth holes unless with a post-processing step to subdivide and smoothen the results, but the simplicity of these approaches makes them a fast solution for some applications such as industrial synthetic foam production.

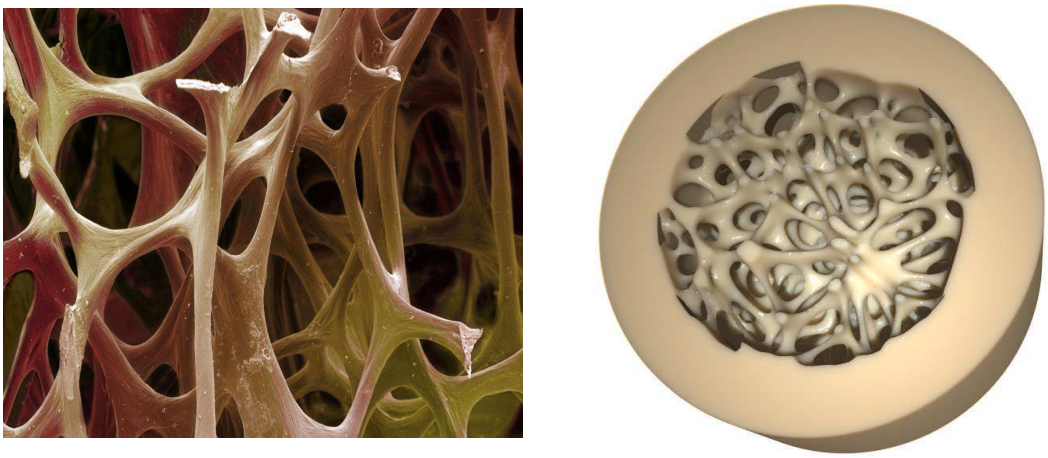


Figure 5.7: A cross-section from Cancellous (spongy) bone inside a simplified Cortical bones of human skeleton (right); a real-life spongy bone (left), credit: Steve Gschmeissner and Getty Images.

5.2 Discussion

Natural modeling of porous or spongy substances is a combination of art and different sciences such as biology, geology, computer graphics, and others. In computer graphics, we use several techniques where each of them helps to mimic one or more characteristics of porous structures to achieve similar results. Our approaches have the capability of constructing several types of these structures by giving the flexibility of choosing and adjusting the right settings for each model. Nevertheless, for a photo-realistic visualization of natural materials an appropriate texture application and different rendering techniques seems necessary as even the best volumetric modeling techniques cannot adequately represent all of the aforementioned examples of porous materials. For our models, we used simple texture mappings and shading techniques such as phong, subsurface scattering (SSS), radiosity, and others, as the goal of this work was introducing volumetric surface modeling approaches. Yet employing more cutting edge methods of 2-D or 3-D textures synthesis, as mentioned in Chapter 2, rendering, and point distributions would be beneficial in our framework.

The computational costs for constructing the figures in this work was a function of different settings. For instance, the fixed or maximum resolution value of 3-D grids in the polygonizer, the method of normal vector extraction, the complexity of the BlobTree which is the the amounts of nodes that it has, and its primitives' functions, and other adjustments in the implementation from the CUDA options to the accu-

racy of feature points extraction in dual contouring. In general, those models whose curvatures were rounder with less thin or sharp features the computation costs were less than 30 seconds, e.g. for a sphere around 2 seconds with 2^7 resolution and for 100 inscribed volumes less than 20 seconds. As the models was getting more complicated, like the spongy bone model, the computational cost was increased, however, we experienced less than 1 minute for most objects. Note that the computational costs do not change linearly with the number of objects of the same type, therefore changing the number of IVs from 100 to 200 does not double the performance time. The performance time for ray-tracing, which is done after the polygonization process, depends on the resolution and the chosen techniques, e.g. the Phong shading is very fast less than 30 seconds to render, but the subsurface scattering with 1000 diffuse scattering samples took around 5 hours. Achieving a real-time interaction with the objects requires considering faster data structures that are compatible with GPGPU implementation. Figure 5.8 shows the polygonization performance, number of points and triangles of some of models in this document.

Model	Time (sec)	Number of Points	Number of Triangles
Triple Torus Object	37	400*3	238210*3
Rocks	55	1200+600	105836 + 83325
Cheese	43	100	305882
Cancellous Bone	73	200	460376
Radiolarian	26	180*2	101229
Pebbles	15	200	192972

Figure 5.8: Comparison of polygonization times for different models.

Chapter 6

Conclusion and Future Work

6.1 Summary of the Contributions and Conclusion

In this work, we presented procedural approaches to creating models containing inscribed volumes. The first step was constructing the 3-D IVs approximated by a set of bounded and convex regions such as Voronoi diagram. The IVs in a negative (inverted) forms can be used for generating porous structures and in their positive (normal) form represents pebbles, natural pearls, modern furniture designs or when the smoothness and aesthetic curvatures are desired, and similar objects. In the second step, we introduced two simple and flexible interconnecting techniques (*fully-connected* and *partially-connected*) to embrace morphologically different spongy structures. The flexibility of our framework gives the freedom of adjusting size and smoothness of the pores as well as choosing a number of chaotic properties for modeling a specific type of sponges where connecting IVs are required. We demonstrated that various natural spongy structures such as Cancellous bones, cheese, rocks, honeycombs, radiolarians can be constructed from our basic isolated IVs or interconnected methods. Synthetic porous materials with uniform pore distributions can also be generated from our techniques with ease as they usually do not have the complex and unpredictable characteristics of natural substances. The results can be deformed or effortlessly blended or combined, i.e. using CSG operations, with other skeletal primitives to construct more complex objects that can be used in different applications such as animations, medical visualizations, or more.

6.2 Future Work

In this work, the definition of IVs was limited to a set of boundaries that were convex and were made of polygonal walls, nevertheless, the IVS can be defined over any bounded convex region as long as the region is presented by its faces and vertices. This means that in a polytope $\mathbb{C}$ curved faces can take over polygon faces. The function must be changed and we have not tested it in our framework yet, but this would be a nice feature which can produce more interesting results. Furthermore, the current implicit function uses the dot product to achieve the distance value for IVS, this restricts usage of non-convex boundaries and the negative values must be discarded from the calculation and it can affect the accuracy of approximated IV. A new mathematical definition is required to overcome this condition while keeping the smoothness properties of the results. This can also help with the interconnected cells as we can assume the neighboring cells as one cell and still produce accurate and smooth IVs.

Although our framework can produce random and uniform control points, as well as seed points generations with specific conditions, it does not use the data from real-life porous materials. Generating *mesoscopic* details of Quasi-Homogeneous materials can be time-consuming on the end-user in the pre-computing step. Integrating a simple and fast texture synthesis procedure or generate various textures and apply them in real-time will reduce the pressure on the user, where he or she can leverage from the combination of our methods and other recent texture synthesis and rendering techniques to achieve the optimal results.

Modeling bubbles are also doable from our proposed method, although it does not a part of our implementation yet, we briefly describe how it can be constructed. In a pack of bubbles, the exterior bubbles are rounder and connected with the neighboring and internal bubbles. For any boundary bubble, if we ignore the shared faces, the IV will be deformed and move toward the shared face and for the internal bubbles, we can consider the biggest inscribed volumes close to the boundaries to create a very thin wall of a bubble. The results can resemble a pack of real-life bubbles.

The Voronoi diagram has been used because of the natural and varied structures it produces. In the future, we plan to investigate alternative tessellation methods so that we can further increase the variety of shapes.

Bibliography

[1] B. Wyvill, P. G. Kry, R. Seidel, and D. Mould, "Determining an aesthetic inscribed curve," in *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. Eurographics Association, 2012, pp. 63–70.

[2] A. Barbier, E. Galin, and S. Akkouche, "A framework for modeling, animating, and morphing textured implicit models," *Graphical Models*, vol. 67, no. 3, pp. 166–188, 2005.

[3] A. Ricci, "A constructive geometry for computer graphics," *The Computer Journal*, vol. 16, no. 2, pp. 157–160, 1973.

[4] O. Gourmel, L. Barthe, M.-P. Cani, B. Wyvill, A. Bernhardt, M. Paulin, and H. Grasberger, "A gradient-based implicit blend," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, p. 12, 2013.

[5] P. Shirley, M. Ashikhmin, and S. Marschner, *Fundamentals of computer graphics*. CRC Press, 2015.

[6] J. Bear and M. Y. Corapcioglu, *Transport processes in porous media*. Springer Science & Business Media, 2012, vol. 202.

[7] J. Bloomenthal and C. Bajaj, *Introduction to implicit surfaces*. Morgan Kaufmann, 1997.

[8] B. Wyvill, A. Guy, and E. Galin, "The blob tree," *Journal of Implicit Surfaces*, vol. 3, 1998.

[9] J. Bloomenthal, "Polygonization of implicit surfaces," *Computer Aided Geometric Design*, vol. 5, no. 4, pp. 341–355, 1988.

[10] B. Wyvill and K. Van Overveld, "Warping as a modelling tool for csg/implicit models," in *Shape Modeling and Applications, 1997. Proceedings., 1997 International Conference on*. IEEE, 1997, pp. 205–213.

[11] B. Wyvill, A. Guy, and E. Galin, "Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system," in *Computer Graphics Forum*, vol. 18, no. 2. Wiley Online Library, 1999, pp. 149–158.

[12] A. Bernhardt, L. Barthe, M.-P. Cani, and B. Wyvill, "Implicit blending revisited," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 367–375.

[13] A. H. Barr, "Global and local deformations of solid primitives," in *ACM Siggraph Computer Graphics*, vol. 18, no. 3. ACM, 1984, pp. 21–30.

[14] M. L. Gavrilova, *Generalized voronoi diagram: a geometry-based approach to computational intelligence*. Springer, 2008, vol. 158.

[15] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[16] C. S. Kaplan, "Voronoi diagrams and ornamental design," in *Proc. Symp. Intl. Soc. Arts, Math., Arch*, 1999, pp. 277–283.

[17] H. Li, K. Li, T. Kim, A. Zhang, and M. Ramanathan, "Spatial modeling of bone microarchitecture," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2012, pp. 82 900P–82 900P.

[18] H. Mayama and K. Tsujii, "Menger sponge-like fractal body created by a novel template method," *The Journal of chemical physics*, vol. 125, p. 124706, 2006.

[19] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, and H.-Y. Shum, "Synthesis of bidirectional texture functions on arbitrary surfaces," in *ACM Transactions on Graphics (ToG)*, vol. 21, no. 3. ACM, 2002, pp. 665–672.

[20] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International journal of computer vision*, vol. 43, no. 1, pp. 29–44, 2001.

[21] J. Filip and M. Haindl, "Bidirectional texture function modeling: A state of the art survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 1921–1940, 2009.

[22] X. Tong, J. Wang, S. Lin, B. Guo, and H.-Y. Shum, "Modeling and rendering of quasi-homogeneous materials," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1054–1061, 2005.

[23] Y. Chen, "3d texture mapping for rapid manufacturing," *Computer-Aided Design and Applications*, vol. 4, no. 6, pp. 761–771, 2007.

[24] S. Magda and D. J. Kriegman, "Reconstruction of volumetric surface textures for real-time rendering." in *Rendering Techniques*, 2006, pp. 19–29.

[25] R. Baravalle, L. Scandolo, C. Delrieux, C. García Bauza, and E. Eisemann, "Realistic modeling of porous materials," *Computer Animation and Virtual Worlds*, 2016.

[26] W. Zheng, J.-H. Yong, and J.-C. Paul, "Simulation of bubbles," *Graphical Models*, vol. 71, no. 6, pp. 229–239, 2009.

[27] O. Busaryev, T. K. Dey, H. Wang, and Z. Ren, "Animating bubble interactions in a liquid foam," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 63, 2012.

[28] J. Martínez, J. Dumas, and S. Lefebvre, "Procedural voronoi foams for additive manufacturing," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 44, 2016.

[29] C. Redenbach, "Modelling foam structures using random tessellations," in *Stereology and Image Analysis. Proc 10th Eur Conf ISS (ECS10)*, vol. 4, 2009.

[30] A. Liebscher, H. Andrä, M. Kabel, D. Merkert, and C. Redenbach, "Modelling open cell foams based on 3d image data."

[31] G. Maliaris and N. Michailidis, "Modeling of open cell structures geometry and mechanical response applying the voronoi tessellation algorithm," in *5th International Conference on Manufacturing Engineering, ICMEN*, 2014, pp. 1–3.

[32] J. F. Blinn, "A generalization of algebraic surface drawing," *ACM transactions on graphics (TOG)*, vol. 1, no. 3, pp. 235–256, 1982.

[33] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura, "Object modeling by distribution function and a method of image generation," *The Transactions of the Institute of Electronics and Communication Engineers of Japan*, vol. 68, no. Part 4, pp. 718–725, 1985.

[34] G. Wyvill, C. McPheeters, and B. Wyvill, "Data structure for soft objects," *The visual computer*, vol. 2, no. 4, pp. 227–234, 1986.

[35] A. P. Witkin and P. S. Heckbert, "Using particles to sample and control implicit surfaces," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 1994, pp. 269–277.

[36] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on visualization and computer graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[37] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proceedings of the conference on Visualization'02*. IEEE Computer Society, 2002, pp. 163–170.

[38] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross, "Shape modeling with point-sampled geometry," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 641–650, 2003.

[39] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge, "Shapeshop: Sketch-based solid modeling with blobtrees," in *ACM SIGGRAPH 2007 courses*. ACM, 2007, p. 43.

[40] H. Grasberger, J.-L. Duprat, B. Wyvill, P. Lalonde, and J. Rossignac, "Efficient data-parallel tree-traversal for blobtrees," *Computer-Aided Design*, vol. 70, pp. 171–181, 2016.

[41] M. Matsumiya, H. Takemura, and N. Yokoya, "An immersive modeling system for 3d free-form design using implicit surfaces," in *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 2000, pp. 67–74.

[42] J. Bloomenthal and K. Shoemake, "Convolution surfaces," in *ACM SIGGRAPH Computer Graphics*, vol. 25, no. 4. ACM, 1991, pp. 251–256.

[43] R. Schmidt, B. Wyvill, and E. Galin, "Interactive implicit modeling with hierarchical spatial caching," in *International Conference on Shape Modeling and Applications 2005 (SMI'05)*. IEEE, 2005, pp. 104–113.

[44] H. Grasberger, A. Weidlich, A. Wilkie, and B. Wyvill, "Precise construction and control of implicit fillets in the blobtree," in *Shape Modeling International Conference (SMI), 2010*. IEEE, 2010, pp. 151–162.

[45] E. P. De Groot, *Blobtree modelling*. University of Calgary, 2008.

[46] R. Vaillant, L. Barthe, G. Guennebaud, M.-P. Cani, D. Rohmer, B. Wyvill, O. Gourmel, and M. Paulin, "Implicit skinning: real-time skin deformation with contact modeling," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 125, 2013.

[47] B. R. De Araújo, D. S. Lopes, P. Jepp, J. A. Jorge, and B. Wyvill, "A survey on implicit surface polygonization," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, p. 60, 2015.

[48] C. Sigg, "Representation and rendering of implicit surfaces," Ph.D. dissertation, Citeseer, 2006.

[49] A. Knoll, "A survey of implicit surface rendering methods, and a proposal for a common sampling framework." *Visualization of Large and Unstructured Data Sets*, vol. 7, pp. 164–177, 2007.

[50] M. Levoy, "Display of surfaces from volume data," *IEEE Computer graphics and Applications*, vol. 8, no. 3, pp. 29–37, 1988.

[51] R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," in *ACM Siggraph Computer Graphics*, vol. 22, no. 4. ACM, 1988, pp. 65–74.

[52] J. Kruger and R. Westermann, "Acceleration techniques for gpu-based volume rendering," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. IEEE Computer Society, 2003, p. 38.

[53] A. Appel, "Some techniques for shading machine renderings of solids," in *Proceedings of the April 30–May 2, 1968, spring joint computer conference*. ACM, 1968, pp. 37–45.

[54] S. D. Roth, "Ray casting for modeling solids," *Computer graphics and image processing*, vol. 18, no. 2, pp. 109–144, 1982.

[55] B. Wyvill and D. Jevans, "Ray tracing implicit surfaces," 1988.

[56] D. Kalra and A. H. Barr, "Guaranteed ray intersections with implicit surfaces," in *ACM SIGGRAPH Computer Graphics*, vol. 23, no. 3. ACM, 1989, pp. 297–306.

[57] G. Wyvill and A. Trotman, "Ray-tracing soft objects," in *CG International'90*. Springer, 1990, pp. 469–476.

[58] J. C. Hart, "Ray tracing implicit surfaces," *Siggraph 93 Course Notes: Design, Visualization and Animation of Implicit Surfaces*, pp. 1–16, 1993.

[59] J.-D. Gascuel, "Implicit patches: An optimized and powerful ray intersection algorithm for implicit surfaces," in *Implicit Surface*, 1995.

[60] A. Sherstyuk, "Fast ray tracing of implicit surfaces," in *Computer Graphics Forum*, vol. 18, no. 2. Wiley Online Library, 1999, pp. 139–147.

[61] I. Wald, H. Friedrich, G. Marmitt, P. Slusallek, and H.-P. Seidel, "Faster isosurface ray tracing using implicit kd-trees," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 5, pp. 562–572, 2005.

[62] E. de Groot and B. Wyvill, "Rayskip: faster ray tracing of implicit surface animations," in *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. ACM, 2005, pp. 31–36.

[63] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.

[64] D. Akio and A. Koide, "An efficient method of triangulating equi-valued surfaces by using tetrahedral cells," *IEICE TRANSACTIONS on Information and Systems*, vol. 74, no. 1, pp. 214–224, 1991.

[65] F. Triquet, L. Grisoni, P. Meseure, and C. Chaillou, "Realtime visualization of implicit objects with contact control," in *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. ACM, 2003, pp. 189–ff.

[66] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature sensitive surface extraction from volume data," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 57–66.

[67] S. Schaefer and J. Warren, "Dual marching cubes: Primal contouring of dual grids," in *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*. IEEE, 2004, pp. 70–76.

[68] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 339–346.

[69] P. Jepp, B. Wyvill, and M. C. Sousa, "Smarticles for sampling and rendering implicit models," *Proc. of 4th Theory and Practice of Computer Graphics (TPCG'06)*, pp. 39–46, 2006.

[70] G. Farin, G. Rein, N. Sapidis, and A. J. Worsey, "Fairing cubic b-spline curves," *Computer Aided Geometric Design*, vol. 4, no. 1, pp. 91–103, 1987.

[71] T. Pal and A. Nutbourne, "Two-dimensional curve synthesis using linear curvature elements," *Computer-Aided Design*, vol. 9, no. 2, pp. 121–134, 1977.

[72] I. Kanaya, Y. Nakano, and K. Sato, "Simulated designer's eyes: Classification of aesthetic surfaces," in *Proc. VSMM*, vol. 2003. Citeseer, 2003, pp. 289–296.

[73] J. McCrae and K. Singh, "Sketching piecewise clothoid curves," *Computers & Graphics*, vol. 33, no. 4, pp. 452–461, 2009.

[74] M. T. Wong, D. E. Zongker, and D. H. Salesin, "Computer-generated floral ornament," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 423–434.

[75] P. Joshi and C. Séquin, "Energy minimizers for curvature-based surface functionals," *Computer-Aided Design and Applications*, vol. 4, no. 5, pp. 607–617, 2007.

[76] H. P. Moreton and C. H. Séquin, *Functional optimization for fair surface design.* ACM, 1992, vol. 26, no. 2.

[77] G. Farin and N. Sapidis, "Curvature and the fairness of curves and surfaces," *IEEE Computer Graphics and Applications*, vol. 9, no. 2, pp. 52–57, 1989.

[78] M. Fontana, F. Giannini, and M. Meirana, "Free form features for aesthetic design," *International Journal of Shape Modeling*, vol. 6, no. 02, pp. 273–302, 2000.

[79] M. I. Shamos and D. Hoey, "Closest-point problems," in *Foundations of Computer Science, 1975., 16th Annual Symposium on.* IEEE, 1975, pp. 151–162.

[80] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1-4, pp. 153–174, 1987.

[81] C. Gold, P. Remmele, and T. Roos, "Voronoi diagrams of line segments made easy," in *Proc. 7th Canad. Conf. Comput. Geom*, 1995, pp. 223–228.

[82] H. Ledoux, "Computing the 3d voronoi diagram robustly: An easy explanation," in *Voronoi Diagrams in Science and Engineering, 2007. ISVD'07. 4th International Symposium on.* IEEE, 2007, pp. 117–129.

[83] H.-H. Hsieh and W.-K. Tai, "A simple gpu-based approach for 3d voronoi diagram construction and visualization," *Simulation modelling practice and theory*, vol. 13, no. 8, pp. 681–692, 2005.

[84] C. H. Rycroft, "Multiscale modeling in granular flow," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[85] C. Rycroft, "Voro++: A three-dimensional voronoi cell library in c++," *Lawrence Berkeley National Laboratory*, 2009.

[86] C. De Boor, "On calculating with b-splines," *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50–62, 1972.

[87] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko, "Function representation in geometric modeling: concepts, implementation and applications," *The Visual Computer*, vol. 11, no. 8, pp. 429–446, 1995.

[88] K. Perlin, "Improving noise," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 681–682.

[89] O. R. Anderson, *Radiolaria*. Springer Science & Business Media, 2012.

[90] K. Rützler, "Family clionaidae d'orbigny, 1851," in *Systema Porifera*. Springer, 2002, pp. 173–185.

[91] B. Julesz, "Textons, the elements of texture perception, and their interactions," *Nature*, vol. 290, no. 5802, pp. 91–97, 1981.

[92] S. Schaefer, T. Ju, and J. Warren, "Manifold dual contouring," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 610–619, 2007.

[93] J. Manson and S. Schaefer, "Isosurfaces over simplicial partitions of multiresolution grids," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 377–385.

[94] B. Wyvill and K. van Overveld, "Polygonization of implicit surfaces with constructive solid geometry," *International Journal of Shape Modeling*, vol. 2, no. 04, pp. 257–274, 1996.

Glossary

Biomimetic The study of structure and formation of living organism structures and materials to produce the imitations of their models and natures for the purpose of human problems. 6

Cancellus bones One of the two types of bone tissues in the human skeleton is called Cancellus bone that has a spongy structure to facilitates the movements of the joints. It is also called spongy or trabecular bone and can be found in skull, ribs, pelvic bones and at the end of long bones. The other type of human bone tissue is called Compact bones. iii

Clionaidae A marine species from the family of demosponges which has boring holes in its structure which results in a three dimensional network of interconnected chambers or galleries below the substrate surface, [90]. 47

Cortical bone A hard and dense outer layer of the bone which are constructed from lots of densely packed cylindrical Osteons. It protects the Cancellous bone and makes up around 80% skeletal mass. 47

Entobia A trace fossil with a hard calcium carbonate surface which has formed from Clionaidae sponges which represents similar network of chamber pores. 48

Laguerre tessellation Laguerre tessellation, aka radial Voronoi tessellation, is similar to Voronoi diagram but instead of measuring proximity between seed points with Euclidean distance, it considers a power distance approach. 8

mesoscopic Of or relating to a scale intermediate between microscopic and macroscopic. 5, 52

polycystines a group of radiolarians that have solid opaline silica skeletons and central capsule which divides the protoplasm into an endoplasm and ectoplasm [89]. 46

radiolarians A type of marine protozoans from the class of Radiolaria. Their siliceous skeletons are usually spherically symmetric with patterns of perforations and often have spicules. iii, 46

spongin A fibrous horny protein that forms the skeletal of marine sponges or Porifera and gives them their flexibility . 1

stereolithography a technique or process for creating three-dimensional objects, in which a computer-controlled moving laser beam is used to build up the required structure, layer by layer, from a liquid polymer that hardens on contact with laser light. 6

texton A term introduced by Julesz in 1981 "the putative units of pre-attentive human texture perception." [91]. In computer graphics, it is referred to the base units of a texture . 5

truss A truss is a structure of individual elements that are connected at their joints where they can also rotate. In truss structure each two element act as axial force at the connected joint while keeping the equilibrium of whole assembly as one object. 6

Appendix A

Implementation

In this appendix, we explain the implementation details of our framework using dual contouring polygonizer, for surface sampling approximation of inscribed volumes which has described in Chapter 4, and the BlobTree.

Many studies on all the contouring methods have accomplished to improve the quality and efficiency of approximated meshed surfaces [47]. Making a decision for the best contouring method for an application depends on various factors such as performance, the necessity for retaining details, and possible limitations such as the type of platform or hardware, the time limitation for implementation, and others. For instance, whether the performance matters or not and if it does, if there are any restrictions on using graphic hardware or not, e.g. the availability or compatibility of an algorithm with an existing GPU card. Although graphics hardware can remarkably increase the performance on many applications, it may not be the best approach for some applications or platforms.

The goal of using dual contouring (DC) in this research was to construct and render the IVs as a single primitive and its combination with any skeletal primitives, whether simple or complex, with smooth or hard edges. We partially used GPU acceleration (NVIDIA GEFORCE GTX 950 and CUDA v.7.5) for our semi-adaptive octree implementation and the most computationally intensive parts benefit from GPU performance.

A.1 Implementation Details

Once a surface is defined by the BlobTree structure in a pre-contouring step, the framework performs contouring to sample the model in a tight finite region around the surface called axis-aligned bounding box (or simply AABB/AAB). In the original DC paper [68], the algorithm relies on readily available Hermite data, i.e. exact edge surface intersection point and normal, that are loaded from an external binary file which can help with the overall performance. Since we are creating a model from one or more scalar field implicit function(s), these Hermite representations must be obtained during grid (regular or octree) traversal for contouring. Apart from extracting scalar fields to perform DC, our framework has deviated from the original DC implementation in several areas.

Most adaptive dual methods for contouring [67, 68, 92, 93] perform adaptive grid tessellation in a bottom-top fashion, meaning that the algorithm subdivides the octree up to sufficiently fine regular grids and then it recursively collapse back to simplify the final contour. In this way, the contoured mesh looks sparser and needs less space for storage. This simplification step can be done prior to contouring, which requires more storage space for a fine polygonal iso-surface, or after contouring. Our implementation uses different approaches to be adjusted for each modeling need. For instance, if a model is a closed surface with relatively minimal details and curvatures, a fully adaptive top-bottom contouring will suffice. This approach can minimize the computational and spatial cost as well as the hassle of complicated restrictive collapsing step when it comes to thin features for flat regions and leaves the surface heavily tessellated. Figure A.1, shows a regular and thin box generated from cube implicit field function: heavy tessellation is avoided while the contouring can sample the sharp areas in a sparse mesh. Although all cubes in this figure are constructed from a fully adaptive grid, the cubes on the top rows do not show this adaptive nature, since the *Quadratic Error Function* (QEF) [68] solver method could sample the exact feature point with minimal error and thus the extra subdivision was not necessary.

For a general purpose modeling, we found that the fully adaptive octree may not precisely sample a detailed surface or a model with dramatic curvature (see Figure A.2). This issue particularly appears when the AABB contains several smaller objects (see Figure 4.11) that are fairly far from each other and the octree cannot robustly estimate the signed edges in the first few levels starting from the root cube. In this situation, our application uses a semi-adaptive or simply regular grids to assure all

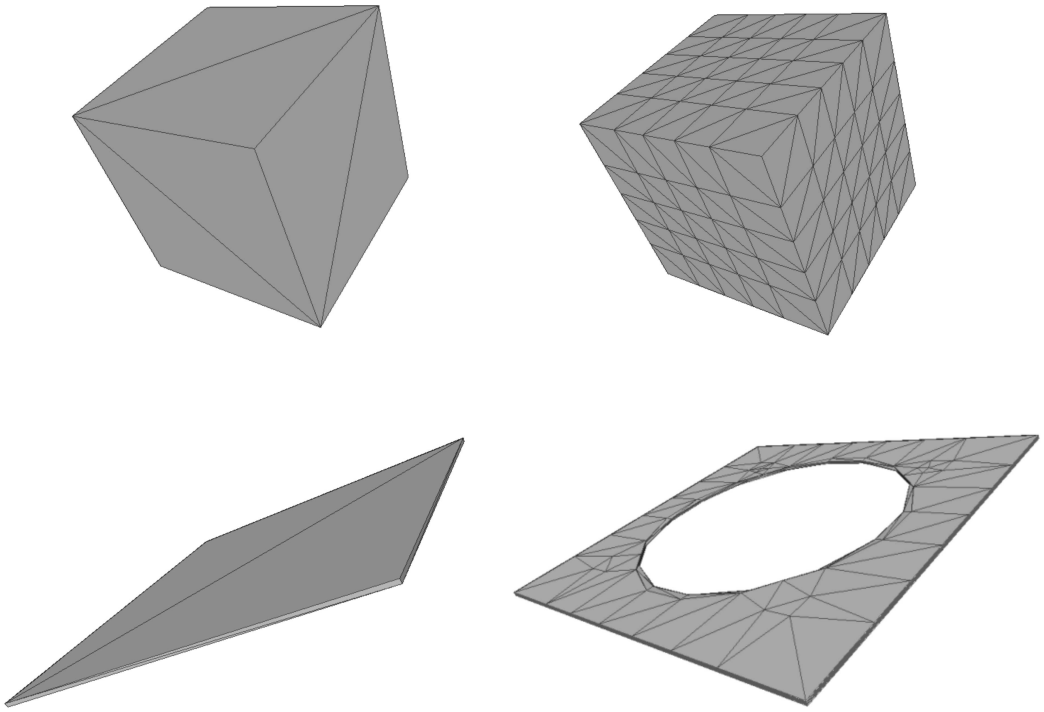


Figure A.1: Adaptive Dual Contouring: the top figures shows a cube surface generated from zero level (top left), and forced to start sampling from level 3 (top right), the bottom row shows a very thin box (1 x 1 x 0.01) (bottom left), and the same box but hollowed by a Boolean operation (bottom right).

the objects in a scene are sampled. An alternative approach is performing contouring method for each individual model in space that is not connected to another model in a scene. We skipped this method for the sake of simplicity, as we treated our blobby inscribed surfaces as a single primitive and since it was mainly used to create spongy structures, we called it a *sponge* primitive. Therefore, even if the sponge primitive was not considered a single primitive and each inscribed blob was supposed to be contoured individually, then performing n-ary operations with other objects from the BlobTree could cause dramatic complications.

A.1.1 Octree Construction

An octree is a tree structure to recursively partition the 3-D space into eight smaller octants (also called cubes, cells, nodes, and voxels). It has been commonly used [68, 92, 93] in adaptive polygonization with dual methods to sample the dual vertices

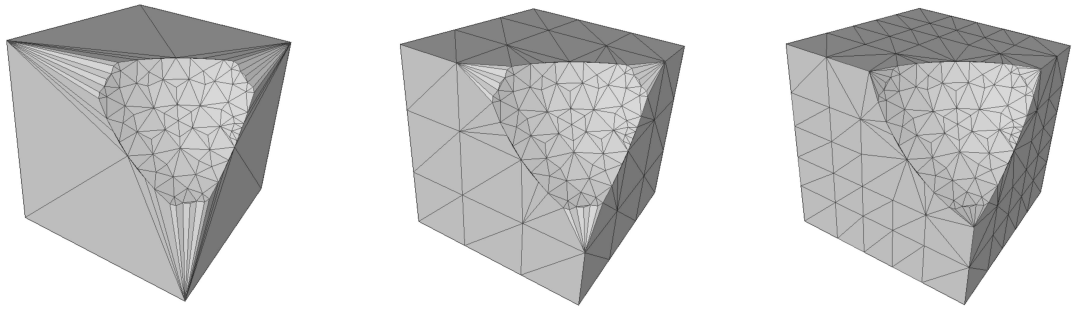


Figure A.2: A non-precise surface approximation caused by fully adaptive Dual Contouring starting from root of octree or level zero (left), semi-adaptive grid starting from level 1 (middle), and semi-adaptive starting from level 2 (right) which results in a precise surface sampling.

(i.e. feature points) and their associated unit normal vectors to indicate the surface orientation. The basic elements of an octree are cubes, faces (i.e. 6 per cube), edges (i.e. 12 per cube), and vertices (i.e. 8 per cube). For each cube, it is sufficient to store only the center position and the size of the cube and extract the rest of elements according to the need.

Adaptive octree subdivision can minimize the memory storage required to store a large number of homogeneous cells in an AABB, as well as refining the sensitive areas, i.e. sharp and or curvy edges, and smaller details, of a surface efficiently. Homogeneous cells are those cubes in the AABB space whose all corner potential field values indicates that they are located on one side of a surface, either inside or outside, and in fact, most of the cells in a regular grid are of this type. On the other hand, heterogeneous cells are the surface crossing cubes by having at least one corner vertex on the opposite side of the surface. Note that we did not use the term *sign* as in some contexts [68] to specify the type of a corner vertex. As has mentioned earlier, we use Wyvill fall-off filter function [5] on skeletal primitives which bounds the field values for each point in space in the range [0,1] and thus a sign change is meaningless. Instead, an iso-value indicates if a point in $\Re^3$ lies inside, outside, or on a surface.

In our implementation, there are four types of octree node: empty (outside), full (inside), internal (has 8 child nodes), and leaf (intersecting). In a top-bottom octree construction, when a node type is declared empty or full, the recursion gets truncated so it is important to tag these nodes with extra care. If a node has intersection with the surface, the error returned from a QEF solver method and the topology of the

intersected surfaces determine if a node needs further refinement (internal node) or is a leaf. As it was mentioned earlier, this method alone may produce a coarse mesh as a result of finding points in bigger size nodes in acceptable ranged errors, where a finer mesh would be more appealing in those areas (see Figure A.3).

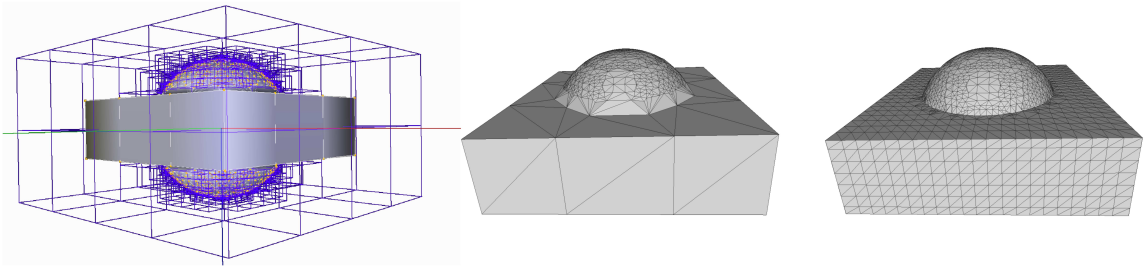


Figure A.3: Adaptive grid cubes in blue lines (left), subdivision stopped at a coarse grid in the intersected part(middle), octree refined up to level 5(right).

In a semi-adaptive octree, space is partitioned up to a prescribed level to assure all objects and their coarse features can be captured by the leaves of the tree. In this step, all the homogeneous cells can be optionally collapsed back to simplify the tree, and a QEF is constructed for the heterogeneous cells to store the feature points (leaf node) and if the error is bigger than a fixed threshold for a cell, it will be tagged as an internal node and adaptively subdivided to extra finer grids or until the error is within an acceptable range, e.g. in our it is implementation 0.0001. The internal nodes are the ones with the potential of having leaf nodes and their types can change during simplification step, e.g. if the all the children of a parent node are full, they collapse and modify the parent node from internal type to full type. The simplification step can be avoided for the simplicity of the algorithm, where the memory storage is not an issue, or can be performed on the leaf nodes containing the feature points. As a result, the octree algorithm has a freedom of choosing the top-bottom or bottom-top manner, or even a combination of both.

The problem with the bottom-top approach is the slow sampling of surface crossing for those implicit functions that are relatively expensive to evaluate since an AABB should be first partitioned into finest grids and all the grid cells should be tested against the surface. In existence of ready-made scene from an external file which contains Hermite data, this may not cause any issue. Nonetheless, for modeling of several objects from scratch, the BlobTree should be traversed for most cells and that means each corner may be evaluated by several functions and operations. A recent method for BlobTree traversal has proposed by Grasberger et al. [40] which

can remarkably improve this performance issue.

GPU Acceleration

Our GPU implementation (using CUDA) is quite simple and helps to accelerate the octree traversed and the BlobTree field evaluations. The recursive generation of polygons for the final meshing is very fast on the CPU, thus it was not required to be implemented on the GPU. CUDA is a parallel computing platform and programming model that helps a compatible GPU to be used in a simpler way by acting as a medium software layer and giving the direct access to GPU's virtual computational functions. The input data are copied from CPU memory (host) to GPU memory (device). The connection between the host and device happen through the *kernel* functions and once a *kernel* is called, the GPU starts parallel computations on each core. There are several third party libraries that can facilitate the data copy and even replace *kernel* functions as well. Once a *kernel* function is called, it gets executed N (*blocks* in a *grid*) times in parallel by M different *threads* (per *block*) (see Figure A.4). The maximum number of N and M are defined in each graphics hardware specifications. For example, our GTX 950m specification shows that the grids are 3-D, and the maximum of *thread-block* for each grid is $(2^31 - 1, 65535, 65535)$, and the maximum of threads per blocks are 1024. In reality, the efficient number of threads that can be specified for a kernel launch also depends on the memory that each thread execution requires, the number of fixed register memory allocation, and other factors. In addition, at each moment, the number of parallel executions cannot exceed the number of GPU multiprocessors by a maximum number of threads per each block.

The three types of functions are: device (specified by *__device__* keyword), host (can optionally specified as *__host__*), and device host (*__device__ __host__*). Since we wanted most functions to be executed by either GPU or CPU, we declared them as *__device__ __host__* functions. As soon as the GPU execution is done and the cached the data on the chip, the kernel function returns, the resulting data are ready to be copied back to host. From CUDA version 6.0 and above, a special type of memory management called Unified Memory (UM) was introduced to dramatically lower the developer's efforts by simplifying the memory access model. It can create a bridge between host and device data transition by creating a pool of shared memory. We used UM for those data that could be shared and modified in both host and device, and thus avoiding the data allocation in both memory as well as copying the resulting

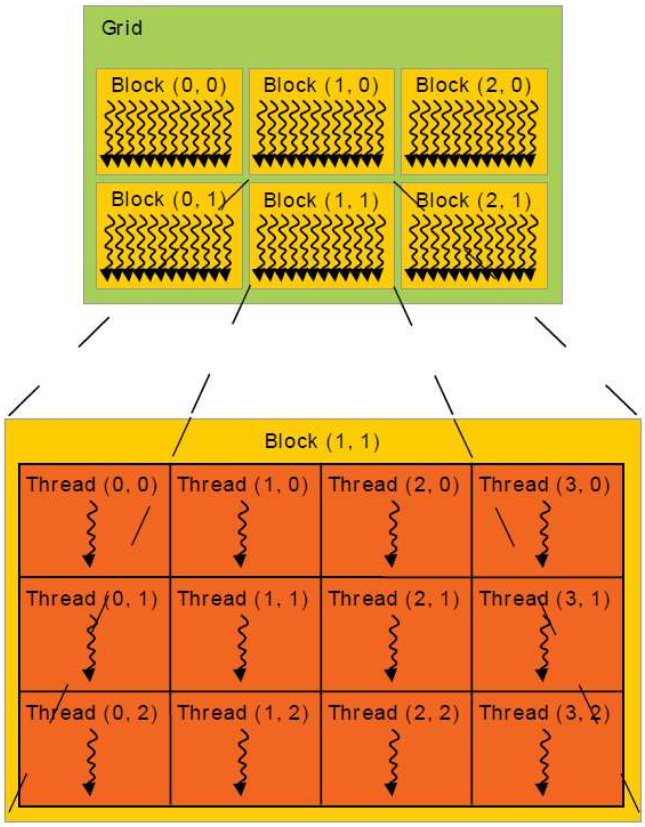


Figure A.4: Grid of thread blocks. Image courtesy of NVIDIA

data between them back and forth.

For a semi-adaptive octree grid, after partitioning AABB up to the desired level, all the leaf nodes (note that the octree leaf nodes here do not necessarily refer to intersecting nodes) are passed to the kernel for parallel evaluations. Each cell is processed independently and first, the corner vertices are examined to classify the type of the cell. If a cell has a potential of missing details of a surface or has a non-manifold topology, it will be tagged as an internal node, otherwise, the QEF can be constructed for that cell. In this step, the majority of the cells will be defined as homogeneous and when the kernel execution is finished and the CPU has access to the cells, these cells can collapse to free the extra memory.

Quadratic Error Functions

Once an octree node classified as a heterogeneous cell, all 12 edges are tested against surface intersection. To find surface crossing points on the edges, a simple linear

interpolation iteration with a fixed threshold and fixed number of iteration, or other numerical methods such as Newton-Raphson, Secant, Bisection, regula falsi methods can be used. For each crossing point, the normal vector of the surface is required to be calculated. The collection of intersected points $\mathbf{p}_i$ and their corresponding normal vectors $\mathbf{n}_i$ is passed to QEF solver class to find a feature point inside each cell. In this method, a point $\mathbf{x}$ is placed inside the crossing cell iteratively to minimize the quadratic function $E[\mathbf{x}]$, where $E[\mathbf{x}]$ is a matrix to solve a linear system of equations $A\mathbf{x} = B$ containing normal vectors (A), and the dot product $\mathbf{p}_i \cdot \mathbf{n}_i$ (B). The quadratic function $E[\mathbf{x}]$ is defined as

$$E[\mathbf{x}] = \sum_i (\mathbf{n}_i \cdot (\mathbf{x} - \mathbf{p}_i))^2 \tag{A.1}$$

Our implementation for solving QEF is similar to Ju et al. method [68]. After solving QEF, there is a chance that the calculated feature point is located outside of the crossing cell which would results in spikes on the surface. A quick fix for this issue is replacing the *mass point* as the feature point. The *mass point* is calculated by the average of intersected points on the edges. This method can help with the spikes on the surface and works well in simpler shapes, however, it can produce non-manifold and inaccurate meshes. It is because the mass point is not always on the surface and forcing one single point inside each cell may not work with some topologies, for example when two parallel parts of the surface are inside of one cell, thus cell subdivision or a safe vertex clustering method [92] must be considered. Note that the non-manifold issue of original DC method does not exists with some other contouring methods such as [94] and [34] or [9], which can be also used for fast surface approximations.

A.1.2 Traversing The BlobTree

Each octree node should be classified as one of the mentioned types to determine if any further process is required. The classification is the cell corners testing step, and as any other points in space, each single corner traverses the BlobTree to query its field value. The BlobTree structure and the operations are explained in Chapter 3 in Section 3.3. To minimize the unnecessary field evaluation, each point is tested against each BlobTree node bounded region and if was not in the region zero value returns. For instance, if two spheres A and B are getting subtracted as (A-B), therefore the

field value of each point outsides of the bounding box of A is zero. The traverse is done in a bottom-top approach by starting to evaluate the leaf nodes functions and until it reaches to the root of the final field value. There are two types of the BlobTree traversal, one is done when it is being constructed and the other is to query field values.

In fact, the BlobTree like any other trees is constructed recursively for a compact structure. However, in this implementation, it is constructed as a linear tree as it has a better compatibility with our CUDA implementation as oppose to parent-child method, i.e. passing the virtual functions to CUDA kernel. Each node is an instance of a *BlobbyNode* class which contains the node data and two Boolean variables to determine if a node has been traversed and whether it is a primitive object or an operation. Figure A.5, depicts the BlobTree nodes construction and the order it is constructed in this implementation: orange squares represents a CSG, blending, and warping operations nodes, green squares are primitive nodes, the ovals are numbered from top to bottom where in each binary node the top child is "inputA" and the bottom child is "inputB", starting from the last tree level. Although blending operation has the ability to be applied over any number of primitives, we simplified the tree to maximum two primitives.

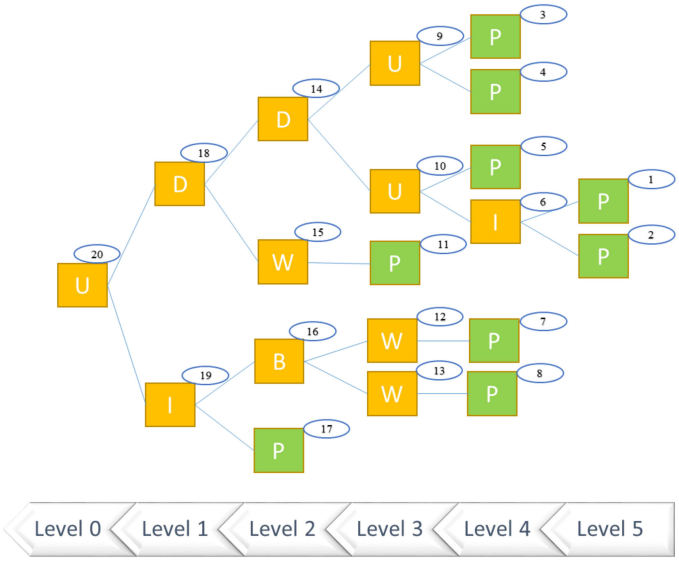


Figure A.5: An example of the BlobTree construction traversal. Green: implicit primitive, orange: binary or u-nary operation.

Once the BlobTree is constructed, it has to be traversed to obtain the field values

for each point. The query traversal starts from the last nodes in the tree and by default, it considers two nodes (A and B) at each time, then it passes towards the root until it finds the first node that is not a primitive (i.e. is an operation) and has not traversed yet. In a case that the founded node (OP) is a unary operation type, the appropriate operation is applied on the last node, with higher index in a list and in this case A, of two nodes and the OP node is marked as traversed and the node B will be analyzed with the next node in the tree. To calculate the normal vector at each point, the normalization of function gradient vector $\nabla f(x, y, z)$ using *finite differences* can be computed since the gradient at each arbitrary point in space is perpendicular to the surface. Using finite differences to calculate gradient vector can be computationally expensive, as it needs six times calculation of implicit function, and will not produce a correct lighting at sharp edges. The reason is because the angle of normal at a sharp point might be considerably different from the rest of a triangle/quad vertices and therefore creates a messy lighting interpolation between the vertices. A simple solution is using the exact normal vector of only one primitive at each node if the node is a binary operation, in BlobTree, which is closest to the queried point and then performing a post processing on the mesh using split normal method (see Figure A.6). The split normal method is an optional step and can be implemented by comparing each vertex normal with the face (triangle/quad) normal and if it is bigger than a certain threshold, it can be replaced by the face normal.

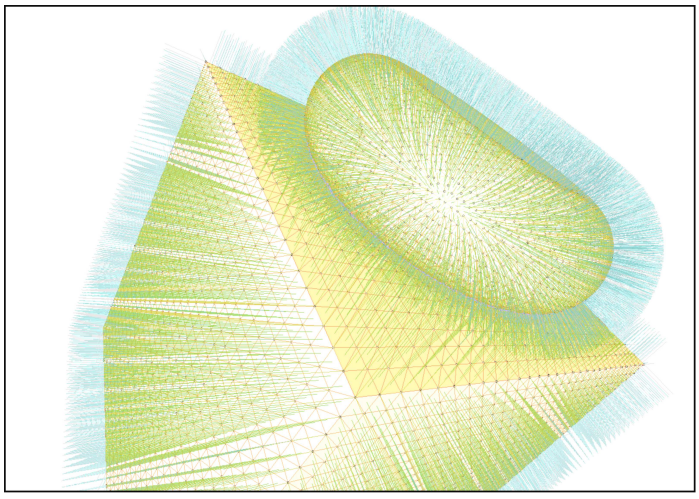


Figure A.6: Split normals at sharp edges to prevent non-appealing lighting effect during smooth shading.

The BlobTree structure helps to recognize the problematic topologies in the DC

as well. Figure A.7 depicts an example of missing surface details with DC method. In this figure, an intersection of two primitives is desired, while the field values of corner vertices represent the voxel is outside of the intersected surface. Using BlobTree structure, it is known that both primitives have intersection with the voxel and their intersection must contain the surface, therefore an appropriate decision should be taken, e.g. subdivision of the voxel, and the same goes for other binary operations.

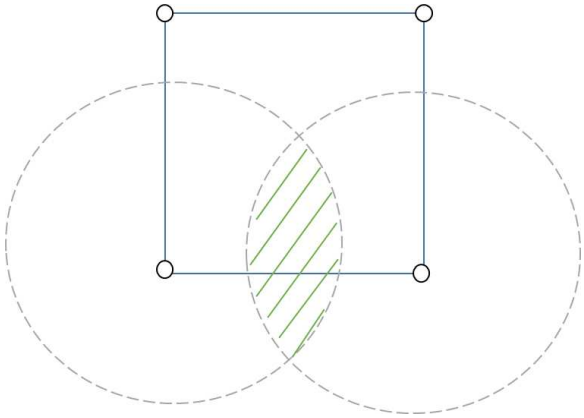


Figure A.7: Intersection of two primitives. The corner vertices field values show that the voxel does not contain the surface.

As mentioned earlier, one of the issues of DC is producing non-manifold meshes in some topologies. Assuming two surfaces are too close to each other such as Figure A.8, the original DC method may find one feature point inside of the voxel while this may lead to producing a non-manifold mesh. This issue can be fixed with a simple test for subdivision: if each two all diagonal vertices of a voxel are nonhomogeneous while a voxel must contain a part of the surface, then there must be a complication in the topology.

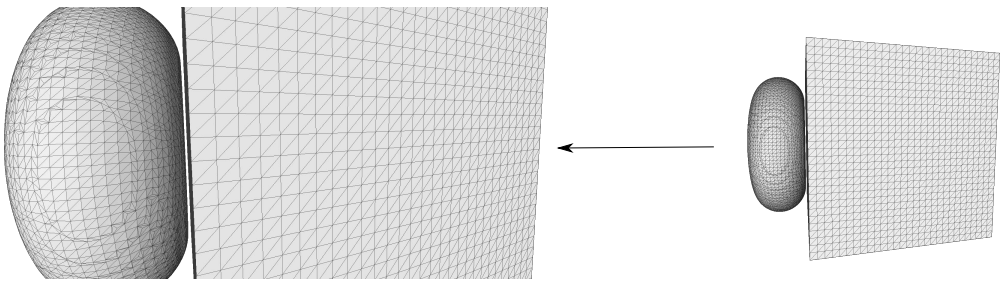


Figure A.8: Two close surfaces are not combined with each other to avoid non-manifold meshes by voxel subdivision.