A GPU Accelerated Signed Distance Voxel Modeling System

Di Zhang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Duane W. Storti, Chair

Mark Ganter

Andrew J. Boydston

Nicholas S. Boechler

Katherine M. Steele

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

A GPU Accelerated Signed Distance Voxel Modeling System

Di Zhang

Chair of the Supervisory Committee:
Professor Duane W. Storti
Mechanical Engineering

This dissertation presents the capability of GPU-based parallel computing to support interactive editing of solids represented by interpolated grids of signed distance values. The basic format of values on a grid is consistent with the format of an image stack that would be obtained from a volumetric imaging system such as magnetic resonance, positron emission tomography (PET), etc. or sent to a variety of 3D printing technology involving digital light projection (DLP) system or Powder bed and inkjet head 3D printing. Thus, this work represents another step toward the goal of a workflow from 3D scan to edit, analyse and fabricate all based on a uniform and robust data format.

While interactive viewing of Signed Distance Field representation(SDF-rep) models has been previously demonstrated, here we present modeling operations that can be achieved in real or near-real time by taking advantage of GPU-based parallelism. Particular operations presented here include importing objects into the SDF-rep modeler, applying Boolean operations, sweeping cut as well as sweeping construction, manipulating the signed distance field by offsetting SDF-rep models (i.e. uniformly manipulating signed distance field to make the original SDF-rep models thinner or fatter without compromising the model integrity). Several applications integrated with those operations above give rise to the capability of modeling a shell version of solids with scaffold structure wrapped inside, which could be essential when printing bio-compatible scaffold material for implanting usage.

In addition, this dissertation also presents a pipeline of computing discrete geometric skeletons from SDF-rep models, editing skeletal data, and refleshing (i.e. computing the distance grid, i.e. solving the eikonal equation, given partial information such as the skeletal data). I further illustrate(and present timings for) the skeletal editing procedures by providing the example of bending a knuckle joint meanwhile adjusting the thickness of fingers on a hand model. Finally, we discuss the essentials of GPU-based parallel implementation and present the report about the efficiency of different options for memory management.

Furthermore, in the body of work, the modeler system is capable for designing multi-material or continuous graded material models by applying user defined material-function on SDF-rep models for image-stack based 3D printing technology. Therefore, continuous gradation material property decoupled from geometry models is realized beyond the limitations of traditional boundary-rep based CAD software. Moreover, examples of creating a spatially controlled materials of various geometries are presented to help understand the whole pipeline.

Last but not least, an approach for reconstructing 3D signed distance field (SDF) model based on surface scanned point cloud data, is introduced here in order to enable the voxel modeler more versatile. Instead of only compatible with volumetric data or image stacks data, the 3D SDF reconstruction is desired for directly importing surface scanned point cloud and creating the signed distance field version of the imported model. This 3D reconstruction method is also discussed in terms of efficiency when employing different amounts of cloud of points or grid sizes. A further discussion about the limitations of our 3D SDF reconstruction is presented to show the robustness of dealing with a few noisy outlier vertexes as well as the failure when importing the non-enclosed surface points data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

PET: positron emission tomography

DLP: digital light projection

SDF: signed distance field

CT: X-ray computed tomography

CAD: computer aid design

AM: additive manufacturing

3DP: three-dimensional printing

CNC: computer numerical control

PMC: point membership classification

MR: magnetic resonance

GPU: graphics computing unit

NURBS: non-uniform rational b-spline

CSG: constructive solid geometry

AMF: Additive Manufacturing File Format

FEM: finite element method

USDF: unsigned distance field or cloud distance

GLUI: GLUI user interface library

GLUT: The OpenGL Utility Toolkit

CUDA: CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA (details in the Appendix A).

ACKNOWLEDGMENTS

I really wish to express sincere appreciation to Professor Storti and Professor Ganter, under whose instruction and financial support, I managed to work on my research topic about the voxel modeler for images based 3D printing. Also, I want to thank my lab mate: Mete, Ben as well as Dr. Marchelli as they are always willing to offer help whenever I meet any problem about the project. With the cooperation with Professor Boydston's lab member Dr. Peterson and Johanna, I learned a lot polymer and chemistry relate knowledge and also were inspired by their passion and fantasy ideas about new material applications. Finally, I want to thank Ricoh, who funded the project and support my software development.

DEDICATION

To my beloved mom and dad

For their endless love, support, and encouragement.

Chapter 1

INTRODUCTION

This dissertation presents a new system for solid modeling and computer-aided design (CAD) formulated to fully support the key advances in fabrication capabilities made available by advances in additive manufacturing (AM), also commonly referred to as three-dimensional printing (3DP). The particular advances provided by AM are: (1) low cost of geometric complexity, (2) mass customization, and (3) continuous gradation of materials and/or properties throughout the interior of the part. To fully support the capabilities offered by AM, a solid modeling system is needed that can: (1) match the resolution of the AM system, (2) readily import scanner data to create digitized objects to which a full set of modeling operations can be applied, and (3) describe variations in properties throughout the interior of an object.

Solid modeling and CAD systems currently available have generally arisen during an era when computer graphics and computer numerical control (CNC) machining were dominant technologies, and both of these technologies focus on surfaces. Computer graphics is driven by simulating how light interacts with the surfaces of objects in a scene. CNC machining involves traditional material removal processes where a blank piece of typically homogeneous stock is fixtured in the machine, and a cutting tool removes material from the blank until it reaches the surface specified by a CAD model of the desired part. In both cases, the focus is on the boundaries of an object, and this focus leads to the boundary-representation or b-rep models composed of a connected collection of surface patches that currently dominate the CAD marketplace. Such b-rep models have several shortcomings when considered in the context of AM: (1) The fundamental function of a solid modeling system is point membership classification (PMC); i.e. distinguishing points that lie in the interior of a model from those that lie exterior to the model. There are significant concerns regarding robustness of PMC

based on b-rep, and very careful bookkeeping is required maintain validity during modeling operations based on b-rep. There are no inherent guarantees that the results of an operation will result is a description of the closed surface of a solid object. (2) While there have been some nominal attempts at describing inhomogeneous objects as a collection of b-rep or describing property variations over the surface, b-rep are simply not designed to describe variations of materials and/or properties through the interior of an object. (3) Import of scanner data to create b-rep of digitized objects remains problematic for both surface scanners and volumetric scanners such computed tomography (CT), magnetic resonance (MR), positron emission tomography (PET), and ultrasound.

To overcome the limitation of b-rep, we have taken an alternative approach based on implicit solid models also known as function-based models or f-rep. The basic idea of f-rep modeling is to define a function over the ambient modeling space that has negative values in the interior region and positive values in the exterior region. An f-rep then provides robust PMC by evaluating the function at a point and classifying the point according to the sign of the resulting function value. Moreover, f-rep of inhomogeneous solids can be readily created by appending additional functions that implicitly define gradations of materials, properties, or processing parameters.

While an f-rep uniquely specifies an object, an object does not have a unique f-rep. Numerous choices are available, and we pursue a particularly useful class of f-rep called signed distance function representations or SDF-rep. In this special class of f-rep models, evaluating the implicit function at a point returns a value whose sign classifies the point (interior vs. exterior) and whose magnitude specifies the distance to surface. Signed distance functions possess several useful properties, and those of particular interest involve the gradient of the SDF which points toward the closest surface point. Wherever the SDF gradient exists, it has unit magnitude; we will see that this is useful for completing partial SDF data. At points where the SDF does not have a well-defined gradient, there is more than one nearest surface point which is indicative of membership in the geometric skeleton or medial axis. We will show that access to the geometric skeleton supports a new and useful set of modeling tools.

We will further present ways to exploit the useful properties of signed distance functions to achieve efficient visualization, preservation of the signed distance property during editing operations, and a practical set of editing tools based on geometric skeletons.

Another major consideration involves dealing with the reality that both scanning systems and AM systems have finite resolution. Thus we choose to employ a finite resolution version of an SDF-rep. Inspired by volumetric imaging systems that produce a stack of 2D images (which can alternatively be considered as a 3D voxel data set), we choose a representation called a discrete signed distance function representation (discrete SDF-rep) consisting of a set of voxel data corresponding to the values of an SDF sampled on a regular grid together with an interpolation method for computing values between grid points.

The description so far involves a discrete, implicit representation of the geometry of a homogeneous solid. We also present the implementation of a straightforward extension of discrete SDF-rep to model inhomogeneous solids that have graded or spatially varying composition and/or properties. We demonstrate the construction of inhomogeneous discrete SDF-rep models by appending to the discrete SDF-rep additional functions (or grids of function values) that define gradations of materials, properties, or processing parameters; and we demonstrate the effectiveness of the modeling approach by presenting test results on graded parts produced via additive manufacturing.

Note that discrete SDF-rep (including the inhomogeneous version) involve image stacks or voxel sets that are not only the native output format for volumetric scanners, but also the native input data required for AM systems such as powder-bed binding and vat photopolymerization where a part is formed (by binding powder particles or solidifying resin) by printing or illuminating a sequence of images. Thus, discrete SDF-rep provide a unified format to support the workflow from 3D scan, to model and edit, to fabrication, and finally to inspection.

This dissertation presents the range of capabilities needed to implement a fully interactive discrete SDF-rep modeling system. It is worth noting that real-time interactivity with large voxel data sets depends on parallelizing intensive computations, and we employ CUDA [2]

to implement parallel computations on the graphics computing unit (GPU) and achieve the desired level of real-time interactivity.

Chapter 2 describes how primitive objects can be created in or imported into the discrete SDF-rep modeler. All the major sources of models are supported including b-rep, SDF-rep, and digitization by volumetric or surface scanning. More specifically, our voxel modeler addresses import of point cloud data typically produced by surface scans. (Segmented volumetric scan data imports naturally into the discrete SDF-rep modeler, continuous SDF-rep can be sampled on a grid to create discrete SDF-rep, and b-rep models can be imported by computing a grid of signed distance values relative to the collection of surface patches; so this represents the significant missing case for importing digitized objects.) We present a straightforward method for computing good approximations of discrete-SDF reps for an object rom a collection of surface points sampled with sufficient density. For clarity, the algorithm is presented along with a 2D example; then results are presented for an actual scan data collected from a 3D object.

Chapter 3 describes how real-time interactivity is achieved by leveraging both the properties of signed distance functions and the power of GPU-based parallel computing.

Chapter 4 presents the basic functions of a discrete SDF-rep modeler including performing the basic Boolean operations (union, intersection, and difference). Since Boolean operations do not fully preserve the signed distance property, methods are presented for repairing discrete SDF-rep by re-computing SDF values in regions damaged by Boolean operations. The basic Boolean operations are then extended to include sequences of geometric transformations to produce SDF-rep models of swept solids.

Chapter 5 presents the classes of geometric operations for creating new geometry with the discrete SDF-rep modeler. In addition to the traditional unary operations (scale, rotate, translate) and binary boolean operations (union, intersection, difference), efficient and robust methods are presented for offset solids, swept solids, and a useful set of operations based on computing and editing a discrete version of the geometric skeleton.

Chapter 6 introduces discrete SDF-rep for inhomogeneous objects create by appending

to the grid of SDF values that define the geometry one or more additional grids describing local variation of materials, properties, or parameters. The chapter concludes with a detailed description of design, fabrication, and testing of octet truss unit cells with graded material properties.

Chapter 7 summarizes the results and presents conclusions that identify the novel and significant contributions presented in this dissertation.

Chapter 2

CREATING AND IMPORTING PRIMITIVE OBJECTS

A significant goal of the discrete SDF-rep modeler is to provide an environment that supports efficient interactions between digital parts designed in a CAD system and parts digitized from scan data. With b-rep modeling technology currently available in the marketplace, these interactions are problematic. Constructing a b-rep from segmented volumetric scan data typically involves marching cubes or related algorithms [3]. Marching cubes produces a large triangle set that definitely sacrifices the ability to represent smooth surfaces and cannot guarantee a result that corresponds to a valid b-rep model of a solid. Constructing a b-rep from surface scan data is also problematic. The clouds typically include a large number of points (tens or hundreds of thousands), and determining how to connect the points to create a triangulation is a combinatorial problem that involves significant computational expense and again sacrifices any chance of representing smooth surfaces. Motivated by the very real need for an environment where digital and digitized parts can interact effectively ("digital" refers objects created in a CAD system while "digitized' refers to objects obtained by scanning"). Therefore, we demonstrate how digital and digitized parts can be imported into the discrete SDF-rep modeler where they can readily interact.

2.1 Discretization of Closed Form f-rep

We consider models created by writing closed-form expressions for SDF-rep. If we start with the simple formula for the Euclidean distance from the origin, $d(x, y, z) = \sqrt{(x^2 + y^2 + z^2)}$, we note that simply choosing a value $d(x, y, z) = r$, or alternatively $d(x, y, z) - r = 0$, provides an SDF-rep for a sphere of radius r centered at the origin. The sphere is imported to the discrete SDF-rep modeler by choosing a grid of points covering a region of space containing

the sphere and evaluating the function $f\_sphere(x, y, z; r) = \sqrt{(x^2 + y^2 + z^2)} - r$ on the grid of points. Note that evaluating a given function on a grid of points is readily parallelizable so that conversion of SDF-rep to discrete SDF-rep is very fast even for the largest possible grids that can be stored in GPU memory (on the order of 1-10 GBytes). While creating other closed-form SDF-rep is a non-trivial endeavor, a number of SDF-rep models are available in the literature [4] including slabs, blocks, cylinders, capsules, and tori all of which import directly to the discrete SDF-rep modeler by parallel evaluation on a grid. Fig. 2.1 shows a visualization of a discrete-SDF rep of some simple geometries derived from a closed-form SDF-rep model.

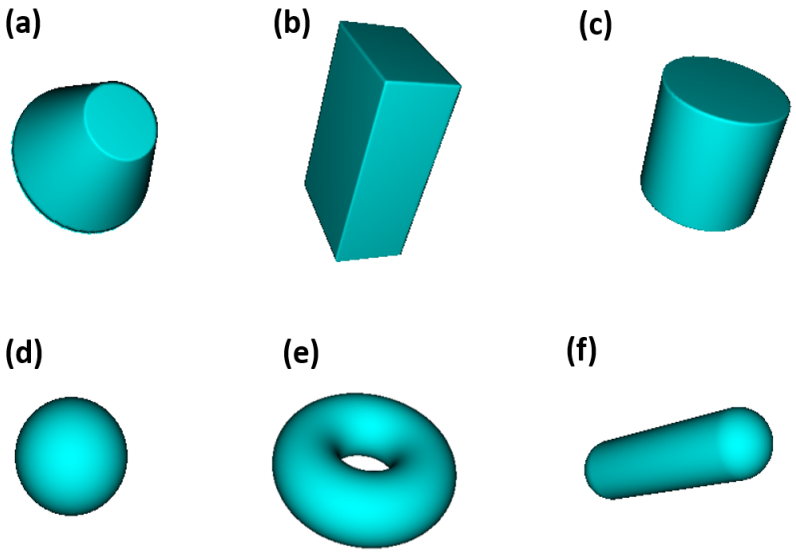

Figure 2.1: 3D signed distance primitives: (a) discrete signed distance cone, (b) discrete signed distance block, (c) discrete signed distance cylinder, (d) discrete signed distance sphere, (e) discrete signed distance torus, (f) discrete signed distance capsule,

2.2 Importing b-rep

Any b-rep modeler that supports point membership classification (PMC) and a surface distance function enables import into the discrete SDF-rep modeler. By placing a uniform grid over the desired object and computing the distance to the surface of the b-rep at each grid point produces a grid of unsigned distance values. Changing the sign at each internal grid point (as determined by PMC) produces the desired discrete SDF-rep model. For polyhedra, the computation involves a sequence of point-to-polygon distance tests which allows parallelization (Ben Wesis's code from 3D printing lab of University of Washington [5]). This approach enables import of the many polygonal b-rep files (STL, PLY, etc.) that are available from numerous sources online like "Thingiverse".

2.3 Import from Volumetric Scans

The first essential step to capturing an object from a volumetric scan involves segmentation [6]; i.e. identifying the voxels in the volumetric scan that belong to the desired object. Segmentation that returns binary (in/out) values produces "old school" or "sugar cube" voxel models that are unsatisfactory because they are incapable of properly capturing information about surface normal direction and curvature. However, segmentation algorithms are now available that provide more refined information. For example, Multirigid [7] implements a two-step approach. The first step uses graph cuts to generate an initial segmentation result. That initial result is smoothed and refined using during an ensuing level set computation that actually computes an estimate of signed distance for voxel centers near the boundary of the object. While the traditional workflow applies marching cubes [3] to produce a triangulation of the zero-distance level set that approximates the boundary, our approach focuses instead on the signed distance values. Since the level set computation provides the signed distance values only in the vicinity of the boundary of the object, the remaining task is to estimate the signed distance values over the remainder of a chosen region of interest.

Fig. 2.2 (a) shows a talus and calcaneous (in the red circle) from CT scanner. Fig.

2.2 (b) shows a segmented talus and calcaneous (in the red circle) among all the other bones with some segmentation algorithm [8, 9], while Fig. 2.2 (c) shows the rendered talus and calcaneous in our discrete signed distance voxel modeler after importing the segmented volumetric data.

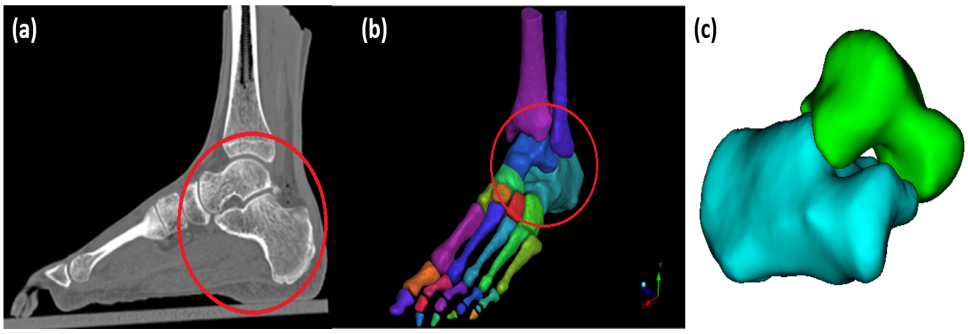

Figure 2.2: CT segmented talus and calcaneous: (a) a talus and a calcaneous are shown in the red circle in the CT scanned image, (b) a segmented talus and a calcaneous are shown in the red circle, (c) a rendered talus and calcaneous in our discrete signed distance voxel modeler after importing the segmented volumetric data

Computing a full SDF grid from values at voxel centers near the boundary brings us to the first instance of completing a partial grid of signed distance values and also the first opportunity of make significant use of the special properties of the signed distance functions. Wherever the gradient of a signed distance function f exists, it has unit magnitude. Thus, the signed distance function satisfies the eikonal equation $|\nabla f| = 1$.

2.3.1 GPU Version of Upwind Differencing for Recovering Signed Distance Field

Our voxel modeler adopts the upwind differencing scheme to solve the eikonal equiaton [10] in Eq. 2.1 in order to create or correct the incomplete signe distance field. Currently, there are three popular approaches for numerically solving eikonal equations, which are fast marching method[11], fast iterative method[12] and fast sweeping method[13, 14, 15], which are essential in many applications including but not limited to medical image segmentation[16, 17],

3D objects reconstruction[18], path planning[19], interface tracking[20, 21]. The eikonal equation, with the relevant boundary conditions, are defined in Eq. 2.1:

$$\begin{aligned} &|\nabla u(x)| = 1,\ x \in \Omega \subset \mathbb{R}^n \\ &u(x) = g(x),\ x \in \Gamma \subset \Omega \end{aligned} \tag{2.1}$$

where u is the unknown value, in our case, replaced by distance value, and f is a given inverse velocity field. g is the value of u at an irregular interface Γ, which could be the enclosed 3D surface or skeleton. There are also varieties of explorations for solving the eikonal equation with algorithm above using serial computing architecture [22, 23, 24]. Here, we report using GPU parallel scheme for solving eikonal equaiotn.

The approach described in this dissertation is based on a parallel fast sweeping method with a nonlinear upwind difference scheme over the entire domain until convergence, which is robust and straightforward to implement. Furthermore, a higher accuracy third order fast marching method for eikonal equation is presented recently[23, 24].

For convenience, the 2D scheme is taken as example here, which is intuitive to extend to 3D space as well [25]. The fast sweeping method employs a Godunov upwind differencing scheme on the updating nodes [26]:

$$\begin{aligned} &[(u_{i,j}^{new} - u_{xmin})^+]^2 + [(u_{i,j}^{new} - u_{ymin})^+]^2 = f_{i,j}^2\, h^2 \\ &i = 2, ..., I-1,\ j = 2, ..., J-1 \\ &u_{xmin} = \min(u_{i-1,j}, u_{i+1,j}),\ u_{ymin} = \min(u_{i,j+1}, u_{i,j-1}) \end{aligned} \tag{2.2}$$

Where i and j are the index along and two orthogonal directions respectively, and h is the grid spacing. The algorithm works by sweeping all the grid points and assigning a new value to u based on Eq. (2.2) replacing previous value if and only if the new value is smaller than previous one. Fortunately, with GPU, threads can be launched parallel, which enables to update each grid point by a single thread without alternatively switching the propagation direction and serial sweeping each grid point within each quarter. Therefore, we can run the parallel upwind-differencing scheme and update the entire grid in less time than you have

for each display update (about 1/60 sec).

In numerically computing world, the detail of solving the Eq. (2.2), is discussed as followed, where at least one of u_{xmin} , and u_{ymin}, exists or stay active (not equal to infinite value).

1. Compare u_{xmin} and u_{ymin}, then determine the smaller neighbor point $u_{smaller}$ as well as u_{bigger}.

2. Solve Eq. (2.2), only considering the contribution term involved with $u_{smaller}$ which could be either the first term or the second term in Eq. (2.2), and get the first trial solution u_{trial} .

3. Compare u_{trial} with u_{bigger} , if u_{trial} is smaller than u_{bigger} , which indicates the u_{bigger} term does not contribute to solving Eq. (2.2). Then, u_{trial} is the final solution to Eq. (2.2).

4. If u_{trial} is bigger than u_{bigger}, then solve Eq. (2.2), considering both terms involved with $u_{smaller}$ as well as u_{bigger} to get the second trial solution u_{trial2} , which indicates that term u_{bigger} does contribute to solve Eq. (2.2). Then, u_{trial2} is the final solution to Eq. (2.2).

The algorithm above is for solving 2D eikonal equation, but extends directly for completion of 3D signed distance grids.

2.4 Import Point Cloud Data from Surface Range Scan

Having already explained how to create discrete SDF-rep models from closed-form SDF-rep, b-rep, and segmented volumetric scan data, we now address creation of discrete SDF-rep models from surface range scan data. The ultimate goal is to realize a system where digital and digitized objects are readily accommodated as illustrated in Fig. 2.3 (discussed in the next section), which shows capability of our voxel modeler for creating SDF-rep models from

various data source. Once we get the capability of dealing with various kinds of data set, it is interesting and worthy showing that the system actually can import, represent, manipulate and even edit on those different data set at the same time. Fig. 2.3 (a) shows a tooth of STL file, which basically is a b-rep model; (b) shows a talus model from segmented volumetric scan data; (c) shows a head model acquired from point cloud, which is the range scan data; (d) shows the union operation (will be discussed in next chapter) is applied between above three models and a torus directly defined with the signed distance function.

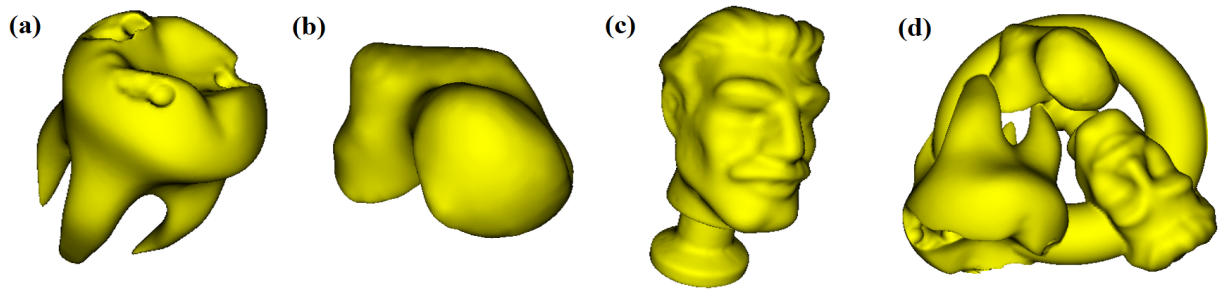

Figure 2.3: the voxel modeler is capable of dealing with different kinds of data set of 3D models: (a) a tooth model converted from STL file, (b) a talus model converted from CT scan, (c) a head model converted from point cloud, (d) applying union operation between previous three models with different data format and a torus directly defined with signed distance function.

This section describes a methodology for importing point clouds and converting them into signed distance field models. A point cloud consists of coordinate values for a set of unconnected cloud points that are assumed to lie close to the surface of an object. File formats for storing point cloud data of scanned objects include XYZ, PCD, etc [27]. Point cloud data is typically acquired from range scanning systems, but for the purposes of constructing examples, the vertices from a polyhedral or mesh model can be extracted using tools such as Meshlab [28]. Since, point clouds themselves are insufficient for the purposes of solid modeling because they do not support a point membership classification for distinguishing

between points that are inside or outside the object. It is desirable to convert the point cloud into a solid model. The traditional workflow for producing a solid model from a point cloud involves determining the topology; i.e. connecting the points to create polygons that collectively form a boundary representation of the corresponding object. This traditional polygonization approach is problematic in two significant ways:

1. Polygonization is expensive in that connecting points into a set of polygons is a combinatorial process that can be computationally costly. If the scanned object consists of multiple disjoint components, additional processing such as K-means algorithm [29] are necessary to cluster the cloud into point sets corresponding to each disjoint component.

2. Polygonization is unreliable in the sense that the resulting set of polygons does not necessarily make up a valid boundary for a solid object.

Here we propose an alternative approach that constructs a discrete signed distance model from a point cloud. As described previously, a discrete signed distance model consists of function values on a uniformly spaced set of grid points. The algorithm involves the following steps:

1. Compute the "cloud distance" grid; i.e. for each grid point compute the distance to the closest cloud point.

2. Determine a distance threshold so that grid points with cloud distance below the threshold create a zone of barrier points that separates regions of internal and external grid points.

3. a) Consider the grid as a graph and, starting from a point on the boundary of the grid, perform a breadth-first search to identify the set of exterior grid points (This step is equivalent to identifying a morphological component in an image); b) Optionally, identify the remaining grid points as internal grid points.

4. Starting with the values on an identified set of grid points (i.e. the cloud distance values on the external points or the negative of the cloud distance values on the internal points), complete the signed distance grid using the upwind differencing scheme (will be discussed in the next section).

5. Optionally, average the two results, apply a smoothing filter, or perform other desired post-processing algorithms.

The algorithm is designed to take advantage of GPU-based parallel computing using CUDA. A significant portion of the computation takes place in steps 1 and 5 which are parallelized as follows:

1. A computational grid was launched with each thread corresponding to a grid point. Each thread computes the distance from its geometric grid point to each of the cloud points and returns the minimum of those distances (In other words, each thread computes the "cloud distance" for a grid point).

2. The signed distance values are stored in CUDA surface memory. Each computational thread updates the signed distance based on neighboring values using the upwind-differencing scheme. It is noticed that the number of iterations is limited by the number of gird points along a coordinate direction, and hundreds of iterations can be executed in a short amount of time.

2.4.1 Computing A Signed Distance Grid From A Point Cloud in 2D

Fig. 2.4(a) shows a cloud of 30 points non-uniformly sampled on the boundary of a "Pacman" region in the plane, and we present figures to illustrate each step in the algorithm for this example. Fig. 2.4b shows the result of Step 1 that computes the cloud distance or unsigned distance field (USDF) for points on the 2D grid.

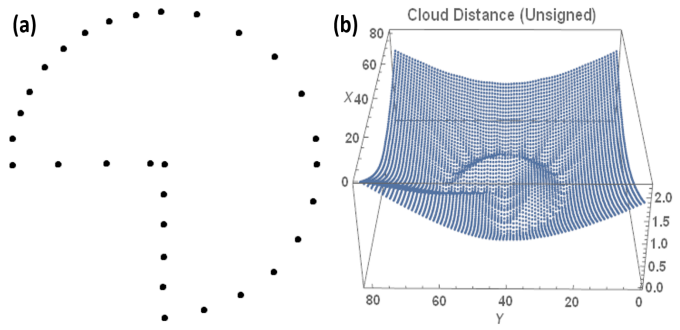

Figure 2.4: 2D data points sampled on the boundary of a "Pacman" region, and its corresponding cloud distance (unsigned distance filed): (a) 30 data points sampled on a 2D "Pacman" boundary, (b) plot of computed unsigned distance field (cloud distance).

Once the cloud distance or unsigned distance field is obtained, we need to determine a distance threshold so that grid points with cloud distance below the threshold create a zone of barrier points that separates regions of internal and external grid points. The initial guess of the distance threshold is the grid spacing, in our 2D case: 0.05. All the grid points whose distance value is below the distance threshold will be classified into a barrier zone, therefore, with distance threshold increasing, the barrier zone is getting thicker. The distance threshold will be increased until the barrier zone is thick enough to separate the internal region from the external region. Fig. 2.5a-c show the zone of the grid where the cloud distance value is below a trial threshold value. Note that in Fig. 2.5 (a) the trial threshold value of 0.05 (grid spacing) is too small to produce an effective barrier zone as shown in corresponding Fig. 2.5 (d), where the external region is still connected with external region. Therefore, the barrier balls' radius keep increasing, as shown in Fig. 2.5 (b), where the distance threshold (or the barrier balls' radius) of 0.10 still can not bridge all the gaps among the cloud points, thus the corresponding Fig. 2.5 (e) indicates the thicker barrier zone does not separate the external region from the internal region yet. After doubling the distance threshold (or barrier balls' radius) again, the next larger threshold values successfully produce a barrier zone as shown in Figures Fig. 2.5(c) with radius of 0.15. Combined with Fig. 2.5 (f), it is

clearly seen that there are three morphology components in the 2D image, indicating that the barrier zone with distance threshold of 0.15 does its job to split the external region from the internal region. Based on establishment of a barrier zone incrementally, we complete Step 2 by choosing a threshold value of 0.15 (shown as Fig. 2.5(c) and (f)) for identification of grid regions.

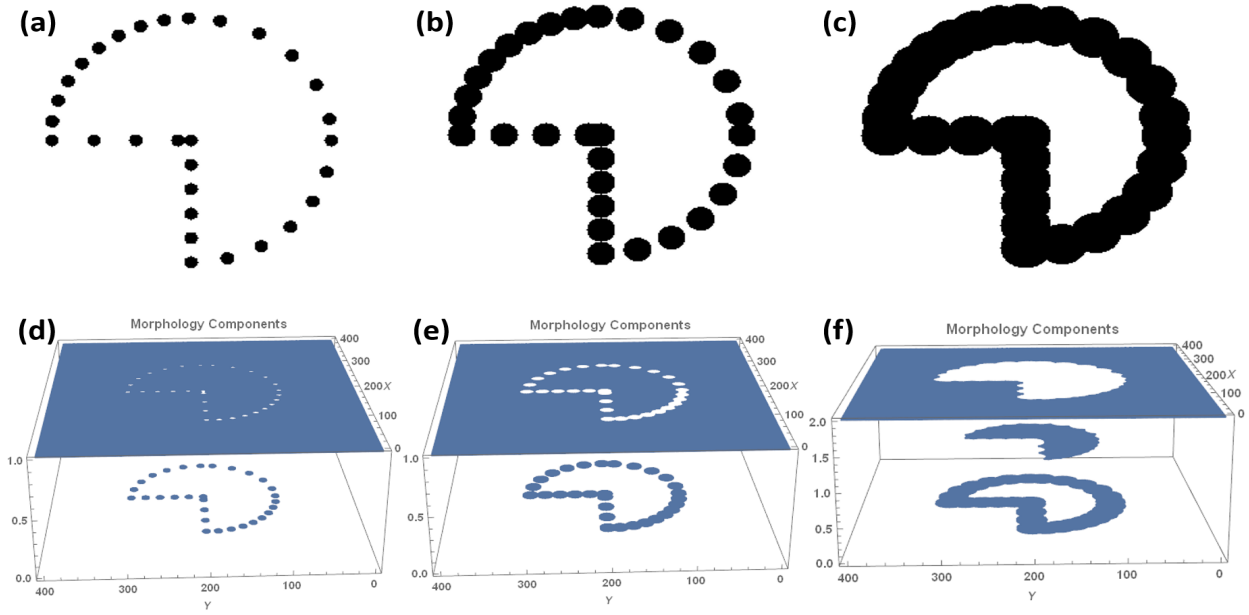

Figure 2.5: Expanded barrier zone on each sampled data point until connected: (a) 30 barrier balls with radius of 0.05 (grid spacing) are located on all sampled data points on its boundary, not connected though, (b) Barrier zone with radius of 0.10 are located on all sampled data points on its boundary, still not connected to separate the internal region from external region, (c) Barrier zone with radius of 0.15 are located on all sampled data points on its boundary, just connected and separate the external region from the internal region, (d) plot of the space with two different morphology components: barrier zone with radius of 0.05 and the rest region (external region is connected with internal region) corresponding to (b), (e) plot of the space with two different morphology components: barrier zone with radius of 0.05 and the rest region (external region is connected with internal region) corresponding to (b), (f) plot of the space with three different morphology components: barrier zone with radius of 0.15, separated external region and internal region corresponding to (c).

Fig. 2.6(a) illustrates the result of Step 4 by showing a plot of the signed distance values for the internal grid points. (Alternatively, Fig. 2.6 (d) shows a plot of the signed distance

values for the external grid points) Fig. 2.6 illustrates Step 4 of the algorithm. Fig. 2.6 (a) shows the initially incomplete signed distance grid with large negative values inserted at external and barrier points. Fig. 2.6 (b) shows the partially completed signed distance grid after 4000 of kernel executions, and Fig. 2.6 (c) shows a plot of the completed signed distance grid(Similarly, Fig. 2.6 d-f show the analogous process starting with known values in the external).

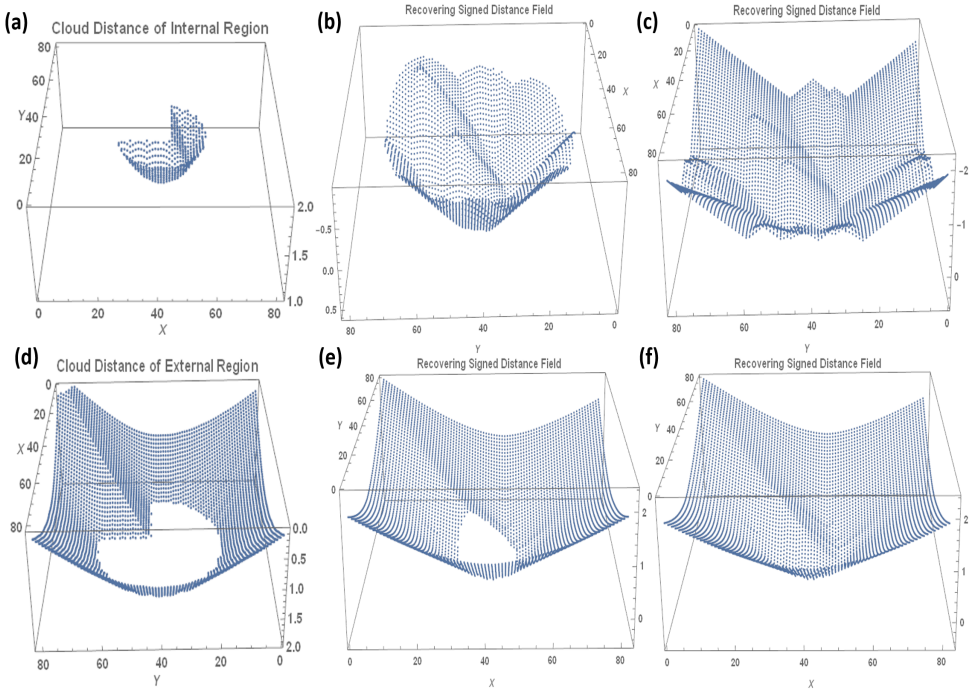

Figure 2.6: Reconstruct signed distance field from internal grid points and external grid points: (a) the top figure shows the original cloud distance of the internal region, (b) shows the partially completed signed distance field based on the initial internal cloud distance, (c) the bottom figure shows the whole reconstructed signed distance field from the inner region, (b) the top figure shows the original distance field of outer region, the bottom figure shows the whole reconstructed signed distance field from the outer region.

Finally, Fig. 2.7(a) shows a plot of the average recovered signed distance field based on the completed internal and external signed distance grids. While Fig. 2.7 (b) shows the

binary picture of a “Pacman”, where the grid points are painted as white if they are negative, vice versa.

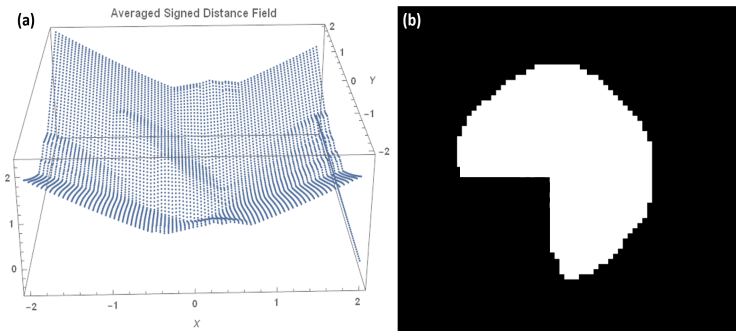

Figure 2.7: the final averaged signed distance field of a circle: (a) the reconstructed 2D circle patch based on the sign, (b) the average of two reconstructed signed distance fields from inner region and outer region above.

2.4.2 Computing A Signed Distance Grid From A Point Cloud in 3D

We now present an example to show how the same idea extends to creating signed distance solid models from 3D point clouds. Similarly, we start by computing the cloud distance first and then adjust the distance threshold to separate the external region from the internal region. The upwind differencing scheme is adopted for recovering the 3D discrete SDF-rep models. it is noticed that this approach produces a solid model without explicitly determining topological connectivity between points in the cloud. A 3D example for showing the idea of thickening barrier is presented in Fig. 2.8. You can imagine there existing a big hole in the point cloud as shown in Fig. 2.8 (a), in order words, there is not any surface point data scanned within that hole. Therefore, just as shown in the cut-away view of Fig. 2.8 (b), it is clearly that, current shell or the barrier zone (the blue one) is not thick enough to form a enclosed region, which further indicates that the barrier zone itself does not succeed to separate the external region from the internal region. As discussed in the previous 2D example, we adjust the distance threshold to thicken the shell or the barrier zone as shown

in Fig. 2.8 (c), however, still not enclosing the internal region. Therefore, we keep thickening the barrier zone until the hole is just filled in Fig. 2.8 (d) and (e). Apparently, although there is a indentation can be seen from outside in Fig. 2.8 (e), it is enough for separating the internal region from the external region. Similarly, the rest is just applying upwind differencing method to recover the whole signed distance field based on the input data of either external region or internal region.

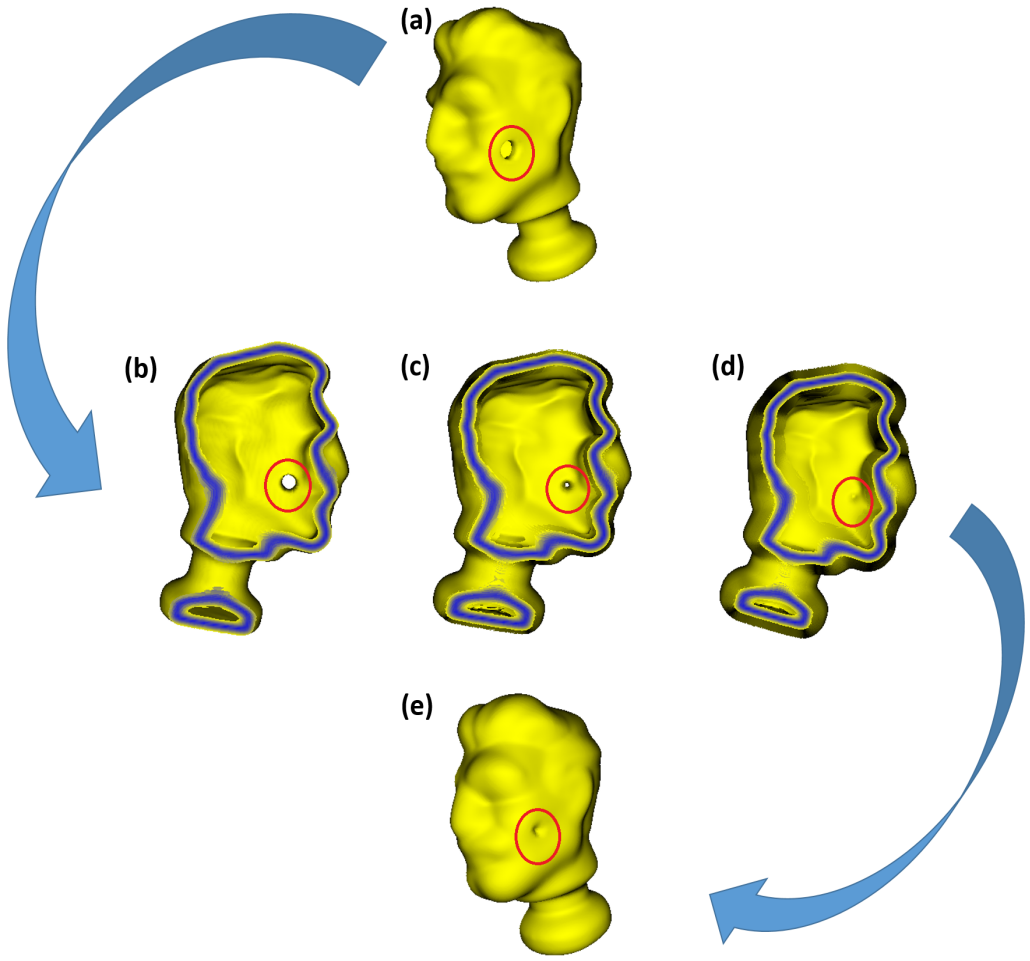

Figure 2.8: increase the distance threshold to thicken the barrier zone: (a) a head model with a big hole on it, (b) a cut through view of the head model, the blue indicates the barrier zone, (c) increasing the distance value to thick the barrier zone, makes the hole smaller, (d) keep increasing the threshold until the hole is filled, (e) a barrier zone, which succeed to separate the internal region from external region, is formed.

Once the 3D signed distance field is separated into external and internal regions, there are two approaches available for recovering SDF-rep models: by choosing either the external region or the internal region as input for repairing the whole signed distance model. Fig. 2.9 shows the images of discrete SDF-rep models for a head recovered from point cloud data. Fig. 2.9 (a) and (b) show the models based on external and internal grid points, and Fig.

2.9 (c) shows the model obtained by averaging the results based on internal and external grid points.

picture (a) is the recovered signed distance head based on the input of external region; (b) is the recovered signed distance head based on the input of internal region; (c) is the average version of previous two recovered versions.

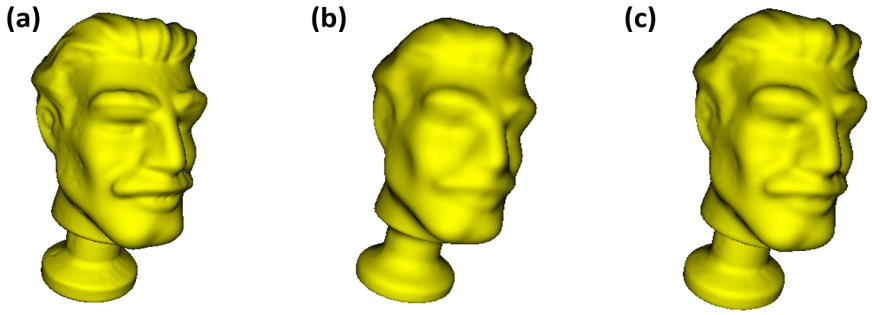

Figure 2.9: 3D signed distance heads: (a)recovered discrete signed distance head from external region (input),(b)recovered discrete signed distance head from internal region (input), (c)averaged discrete signed distance head.

It is interesting to see scalability of our algorithm, since it is important to see whether it is efficient or not when dealing with large data set. The experiments below discuss the efficiencies when dealing with different resolutions of input surface points data as well as voxel grid sizes. The first experiment conducted is to time the whole 3D reconstruction procedure with different voxel grid sizes from 32*32*32, to 64*64*64 and even 128*128*128, while keeping the same amount of surface data points as 85442. However, it does not mean we should always use the large voxel grid size since it might be really computational intensive. And the timing experiment result shown in the Table 2.1 indicates that the 3D reconstruction procedure consists of three parts: we first compute the cloud distance or unsigned distance field (USDF) without classifying the points membership, thus all the distance values is positive at this stage; then, we use the distance threshold adjustment trick to expand the barrier zone until it successfully separate internal region from external region

(point membership classification). During this step, we start with one fifth of grid spacing as initial distance threshold, and keep doubling it until the threshold can form a barrier zone separating the external region from internal region. Finally, the upwind differencing method is applied on the data from the internal or external region to complete the signed distance field. Among those three parts, USDF computing and SDF repairing are executed in parallel on GPU, while point classification by adjusting the distance threshold to form a barrier zone, involves a CPU-based serial implementation (naive serial depth first search adopted here). From the Table 2.1, we can see that with number of grid points increasing, the USDF computing and point classification parts take longer time, which contributes to increasing total time. Besides, it is noticed that the points classification part takes most of the total time, usually weighting around 77% in the total time during 3 steps of 3D reconstruction (USDF computing (parallelized), points membership classification (serialized), recovering the SDF (parallelized)).

Table 2.1: Performance of reconstructing the 3D tooth from 85442 surface data points with different voxel grid sizes: 32^3, 64^3, 128^3 respectively.

Voxel grid size	USDF computing	Point classification	SDF fixing	Total time
32^3	475.8 ms	782.5 ms	2.0 ms	1259.8 ms
64^3	2168.7 ms	6660.3 ms	8.3 ms	8837.5 ms
128^3	16495.7 ms	57183.2 ms	71.1 ms	73750.2 ms

Instead of increasing the voxel grid size for representing the signed distance field 3D model, it is equally interesting to have a look at the performance when we dealing with different amounts of surface data points. The second experiment conducted is to check the performance of 3D reconstruction procedure based on different amounts of surface data points, from 42721 to 85442, and 170882, while keeping the same voxel grid size of 128*128*128.

Similarly, the timing experiment result shown in the Table 2.2 indicates that with the number of surface data points increasing, the cloud distance or USDF computing time in-

creases correspondingly from 8237.7 ms to 16495.7 ms, 33061.9 ms , but the point classification part takes almost the same amount of time as 57189.6 ms, 57183.2 ms and 58766.4 ms. From the Table 2.2, it is further shows that increasing the amount of surface data points only has small effect of increasing the total computing time for 3D reconstruction compared with increasing the the number of grid points. Because the surface data points are only used during the unsigned distance field (USDF) computing in parallel, where a very nearly linear relation between the number of cloud points and time for computing partial SDF can be seen in the Table 2.2 that the first column of USDF computing time: 8237.7 ms, 16495.7 ms, 33061.9 ms is increasing correspondingly to the increasing number of surface data points from 42721, to 85442 and 170882. Nevertheless, the increased USDF computing time does not affect the total time dramatically since point classification is always the bottleneck of this algorithm. Therefore, the time complexity of our algorithm is always O(N) in terms of number of input data points, instead of O(N*N) compared with the time complexity of Crust algorithm [30], figuring out topology generation among all pairs of points and poles.

Table 2.2: Performance of reconstructing the 3D tooth from 42721, 85442, 170882 surface data points respectively, all represented by voxel grid size of 128^3.

Num. of points	USDF computing	Point classification	SDF fixing	Total time
42721	8237.7 ms	57189.6 ms	71.4 ms	65498.1 ms
85442	16495.7 ms	57183.2 ms	71.7 ms	73750.2 ms
170882	33061.9 ms	58766.4 ms	70.1 ms	91898 .9 ms

2.4.3 Limitations of The Signed Distance Filed 3D Reconstruction Algorithm

It is interesting to test some boundary cases of our SDF 3D SDF-rep models reconstruction algorithm. The first scenario is what happens if there is a big hole on point cloud (not a closed surface). In order to conduct the experiment, a STL file of a dinosaur is shown in Fig. 2.10 (a), where it is seen that there is not any triangle mesh or vertex on the bottom

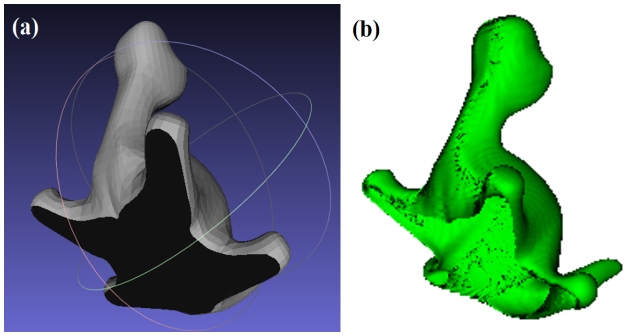

Figure 2.10: An example of 3D dinosaur reconstruction from the non-enclosed surface point cloud data: (a) is STL model of the non-enclosed dinosaur, (b) shows the reconstructed signed distance field shell.

part, which contributes to the black color in STL visualization. With our 3D reconstruction algorithm, all the vertexes are imported as surface point cloud data. After keeping increase the distance threshold, our system reconstruct the 3D SDF-rep model shown in Fig. 2.10 (b), where, it produces a shell (not a enclosed surface) instead of an enclosed solid model.

Another interesting topic is: what if the inner regions are not connected or there are multiple disconnected inner regions. For clearer explanation, a simple example of three disconnected spheres are shown in Fig. 2.11. Fig. 2.11 (a) shows an example involving a scan of a region containing three disconnected objects: a torus, a hand and a cone capsule. The result of applying the SDF-rep import algorithm is shown in Fig. 2.11(b). Note that a model consisting of the union of the scanned objects is obtained without the need to cluster data points into connected components.

To investigate the effect of outlier points, we added two outlier points to the scan data for a set of three disjoint spheres as shown within red circle in Fig. 2.12. While the outlier points affect the cloud distance values for nearby grid points, all such points are external to the solid. Thus the SDF-rep obtained by completion of the data associated with the internal grid points is free of artifacts due to the outlier scan points.In the worst case scenario of outliers both external and internal to the object, discrepancies between the completed SDF-rep based on intenal and external data identify regions for possible user intervention.

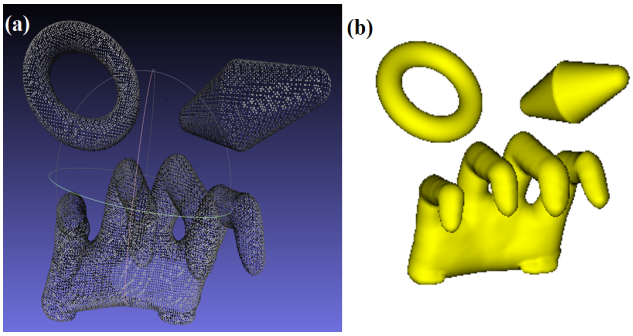

Figure 2.11: An example of reconstructing three disconnected spheres: (a) shows the surface point cloud data of three disconnected objects, (b) shows the separate reconstructed SDF objects.

Successful SDF-rep import of the model of three spheres despite the presence of outliers in the scan is illustrated in Fig. 2.12 (b) and (c). Although, the outlier vertexes affect the cloud distance values for nearby grid points, causing some voxels possessing different distance value to the outliers instead of the closest surface points. However, we can easily rule the outliers out by simply applying the upwind differencing for repairing the whole signed distance field: firstly, we recover internal distance data using partially correct external distance data. The correctness of internal signed distance data is ensured due to the fact that merely outliers nearby grid points are affected, while the voxels close to the surface (used as boundary condition to recover the internal distance data) keep the correct distance value; Then, we run the SDF repairing algorithm, using the internal distance data as input to repair external voxels, where the outliers can be ruled out. After running our 3D SDF reconstruction algorithm, the outliers are not visualized in our SDF modeler as shown in Fig. 2.12 (b). However, visualization might be misleading sometimes due to resolution, therefore, in order to further check the reconstructed 3d signed distance field, we further check the image stacks output by our system, shown as the Fig. 2.12 (c). It is clearly seen that there is not any internal voxel or voxel close to boundary mis-classified around the location of previous outliers, which further proves that a few outliers will not affect our reconstructed 3D signed distance field.

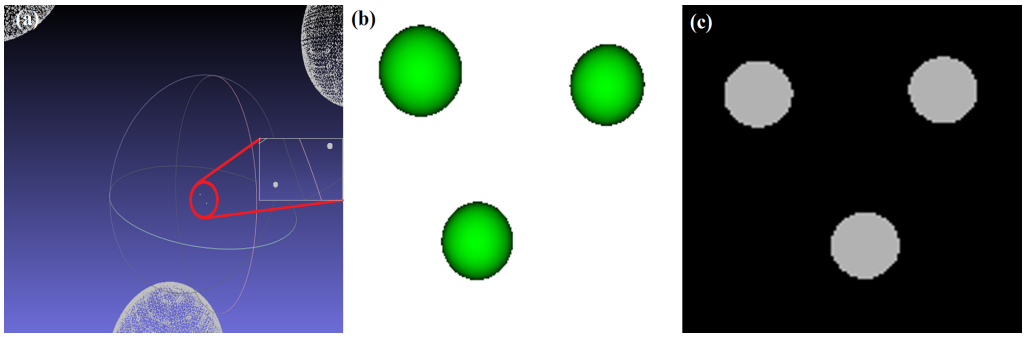

Figure 2.12: An example of impact of having several outlier vertexes on our SDF 3D reconstruction result of three spheres: (a) shows the two manually added outlier vertexes, (b) is our reconstructed 3D three spheres without the effect caused by the outlier vertexes, (c) shows the image stack our system eventually outputs for image friendly 3D printing.

Chapter 3

BASIC SDF-REP MODELS INTERACTIONS AND VISUALIZATION

This chapter describes how real-time interactivity is achieved by leveraging both the properties of signed distance functions and the power of GPU-based parallel computing.

3.1 CUDA Graphics Interoperability

Realization of a fully interactive SDF-rep modeler depends on being able to compute, modify, and visualize signed distance data sets of non-trivial (1GB) size at interactive rates (on the order of 60 frames per second). The goal of interactive SDF-rep modeling depends on both large computing throughput and rapid availability of results for display. We achieve those dual criteria by taking advantage of the interoperability between CUDA and OpenGL, often abbreviated as "CUDA/OpenGL interop". CUDA [2] is NVIDIA's combined hardware/software platform for parallel computing on the GPU, and OpenGL is a standard application programming interface (API) for computer graphics. CUDA/OpenGL interop enables allocation of memory on the GPU that can be readily mapped back and forth between CUDA (which computes results to be displayed and stores them in the allocated memory) and OpenGL (which displays the results in a graphics window and provides support for basic user interactions).

Generally speaking, there are three steps for setting up compute/graphics interoperability: (1) setup the objects in the graphics context, (2) register objects with the compute context, (3)map/unmap the objects from the compute context.

CUDA/OpenGL interop eliminates the need for memory transfers for purposes of display. When the memory is mapped to CUDA, computational results can be stored as usual for

global memory on the GPU. When mapped to OpenGL, the same data is used as the basis for display in a graphics window. The specifics of implementing CUDA/OpenGL interop involves the following function calls: (1) $cudaGLRegisterBufferObject(GLuintbuffObjb)$ is called to register a buffer with CUDA (used for transfer data to OpenGL later); (2) $cudaGLMapBufferObject(void**devPtr, GLuintbuffObj)$ is called to map the buffer object to memory, which actually returns an address in previously registered global memory buffer [31]; (3) a cuda kernel is launched to process the buffer (determine what color should be displayed on each pixel); (4) unmapping the buffer object prior to use by OpenGL for rendering later with the function $cudaGLUnmapBufferObject(GLuintbuffObj)$; (5) finally, the buffer object is used by OpenGL code for display.

Once the basic code is in place to display an array of shading values on the screen using CUDA/openGL interop [32], developers can focus on programming the kernel to perform parallel computation of desired shading values. In our voxel modeler, the computation is divided into a 2D computational grid in which each thread is responsible for computing the shading value for a pixel in the image to be displayed. The sections below will talk in detail about how we actually determine the color for each pixel on a 2D window. In addition to rendering, our discrete SDF-rep modeler, like all CAD systems, needs to support interactions with the user. While OpenGL enables handling user input, we use GLUI [33] to build the user interface and to specify the actions associated with input from mouse and keyboard.

Basically, GLUT is an OpenGL Utility Toolkit [34] and GLUI is a GLUT based C++ user interface library which provides controls such as buttons, check box, radio button and spinners to OpenGL applications. Features of the GLUI user interface Library include (from GLUI manual by Paul Rademacher [35]):

1. Complete integration with GLUT toolkit

2. Simple creation of a new user interface window with a single line of code

3. Controls can generate callbacks when their values change

4. Variables can be linked to controls and automatically updated when the value of the control change (live variables)

5. Controls can be automatically synchronized to reflect changes in live variables

6. Controls can trigger GLUT redisplay events when their values change

Fig. 3.1 shows the user interface of our voxel modeler created with GLUI library. The window visualizing SDF-rep models appears on the left, and the user controls appear in the panel on the right.

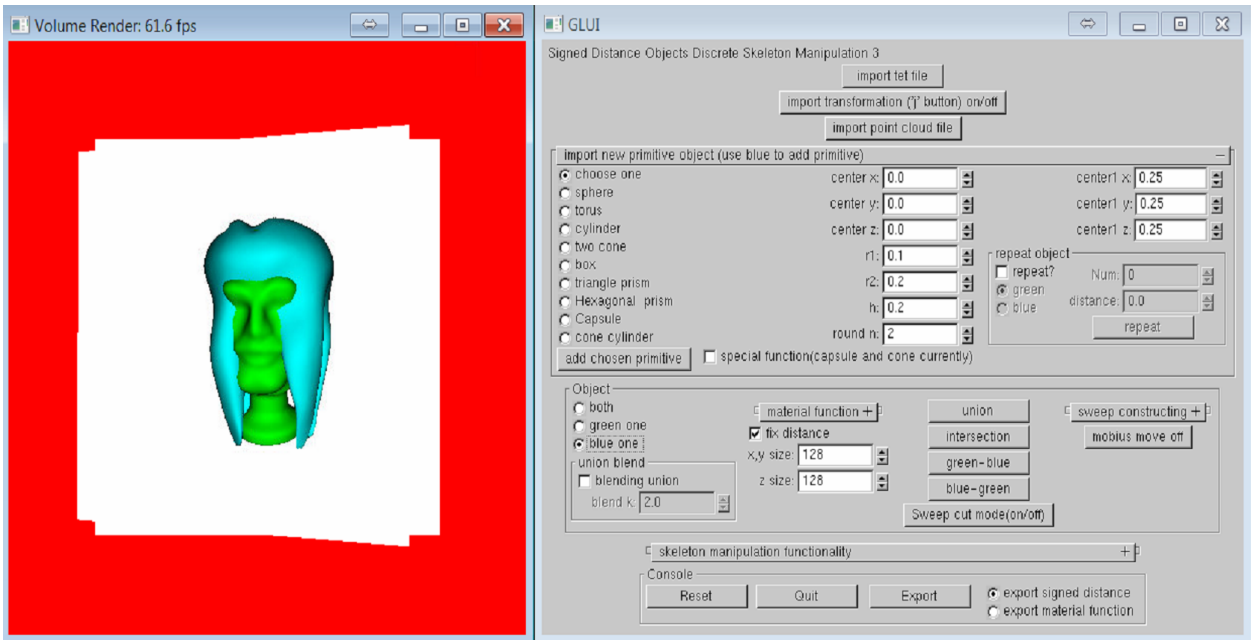

Figure 3.1: GLUI user interface: the window visualizing SDF-rep models appears on the left, and the user controls appear in the panel on the right.

Besides, GLUI provides users with the capability to interact with the system with keyboard and mouse commands. It detects the key users press and hold and then triggers the

callback function to control the system. Similarly, with the capability of keeping track of the mouse movement, GLUI enables the users to transform the models by just clicking and dragging in the designated direction.

3.2 Ray Casting Render of Discrete SDF-rep Models on The Desired Level of Real-time Interactivity

Essentially, all the visualization issues are related to the problem of root-finding, for instance, in polygonal objects visualization, the root finding of any defining function with respect to the spatial coordinates is achieved through looking for nearby point on the surface of objects. Here, it is indicated that the visualization of signed distance solids with root-finding is more efficient and convenient using an alternative approach called ray-casting. A similar volumetric rendering scheme was reported[36] and with GPU CUDA parallel scheme[37]. Basically, ray-casting inverts the real world physics of vision by merely back tracing those lights does get into the eyes from the source, as a result, we avoid computing those lights from the source but never ever get into the eyes of viewer. The procedure of root finding based on ray-casting in signed distance field is straightforward by sampling the step size, the step size equals the signed distance value sampled or interpolated among certain grid points. Besides, ray casting method not only provides the way to find roots on the surface of objects, but also offers a simple and practical approach for getting shading intensity by computing the angle between the ray direction and the outward surface normal.

The ray intersection follows the procedure in detail: A parametric ray shooting from the viewpoint in 3D space is determined by associating the "eyes" position with the corresponding pixel position on the view port window, which is located between the 3D objects and the "eyes". The 3D signed distance solids are scaled in a bounding box centered at origin in the world coordinate. Instead of directly computing the intersection with the signed distance objects, the algorithm first returns the parameter of the ray intersecting with the bounding box since it can scale down the problem without further considering the rays which even never intersects with the bounding box. On the other hand, the ray intersecting with the

box still may or may not intersect with the objects. Therefore, the ray casting approach needs to check whether it is hitting the objects. With the method of sampling distance value as step size, the root finding is efficient compared with taking constant step size. Along the ray casting direction, once the sampled distance value is within certain threshold away from the zero set surface of the objects, the ray is determined as intersecting with objects. Then the shading intensity is computed based on the angle between this ray direction and the outward surface normal. In addition, the threshold adjusting can provide us a way to achieve rendering the signed distance field representation based objects on different sets of surface in users' interactively real time. What is more, with our signed distance modeling system. This capability of adjusting the offsetting surface can be extended to make SDF models thinner or fatter compared with the zero set surface models, which, however, could be slow or even problematic for traditional polygonal modeling system.

For convenience, a 2D example is provided here in detail, whose concept, however, is straightforward to extent to 3D objects as well. In the 2D example of a disk, the contour plot indicates the distance away from the boundary of the 2D SDF circle as shown in Fig. 3.2(a). If the view point is o, and the ray oa is not even interesting with the square bounding box, then this ray is abandoned for further consideration of intersecting with the disk. Only those rays which intersects with the square box, are considered as potential rays further intersecting the disk. As shown in the Fig. 3.2(a), the red points of ray oc and ob indicate the location of each step, and the distance value sampled by the red points on their location are considered as the next step size, which is actually numerically equivalent to the Euclidean distance from the red point location to the closest boundary of the disk. Eventually, the ray ob takes five steps until it leaves the domain without intersecting with the 2D disk in the middle since none of the distance values sampled by those five red points falls within the threshold considered as intersecting the circle's boundary. Up to this point, it is more interesting to observe that the ray oc is interesting with the disk. In this case, each step size it takes is roughly the real distance away from sample location to the boundary of the disk, thus only two steps need to take before hitting the disk, which contributes to an much

more efficient marching procedure to find intersections between the ray and the object's boundary, compared with solving implicit equations to find the roots for the mesh objects. As shown in Fig. 3.2(b), to help understand how big the step size it needs to take whenever marching along the ray at different locations, the 3D plot of 2D signed distance disk is provided corresponding to the contour plot, where, the z axis indicates the distance value or the step size sampled on each grid points.

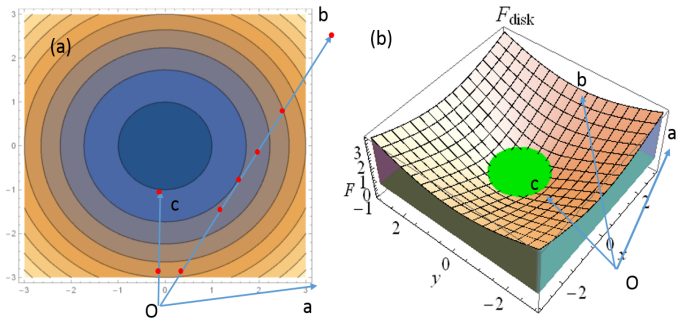

Figure 3.2: 2D signed distance disk visualization rendering: (a) contour plot of 2D example of a signed distance disk, (b) 3D plot of 2D signed distance disk.

For 3D signed distance objects rendering, it also shares the same mechanism as 2D example presented above. Even though for 3D signed distance field with more grid points, we can still achieve the root-finding within reasonable amount of steps taken for rendering the signed distance objects. Taking the example of a signed distance sphere with radii of 30 in a bounding box with side length of 128, as shown in Fig. 3.3(a), the shading is based on the angle between the ray direction and surface normal direction. In addition, for further explanation about efficiency when marching along ray, the Fig. 3.3(b) associates the brightness with the amount of steps each ray needs to take. Basically, those rays not intersecting with the bounding box contributes to the pixels outside the box, which are always green. However, inside the bounding box, it is shown that there is different brightness distributed, where, darker black indicates fewer steps before it hitting the sphere or penetrating bounding box without hitting anything, while brighter white indicates more steps no matter whether

it hits the sphere. The range of steps in this figure is 0 to 15 steps, which illustrates efficient root finding for intersections. Only those rays closely passing by the sphere but not really hitting the sphere march more than 15 steps, thus, in the Fig. 3.3(b), rending the tiny area around the sphere as bright white. Those rays marching with relatively many steps actually can be easily limited by setting maximum steps. This tiny bright area around the circle makes sense since those rays passing by the boundary sample the tiny distance value as their step size, however, marching in the direction which would never intersect with the sphere, therefore, those rays contribute to many small steps, which are rendered as bright white area.

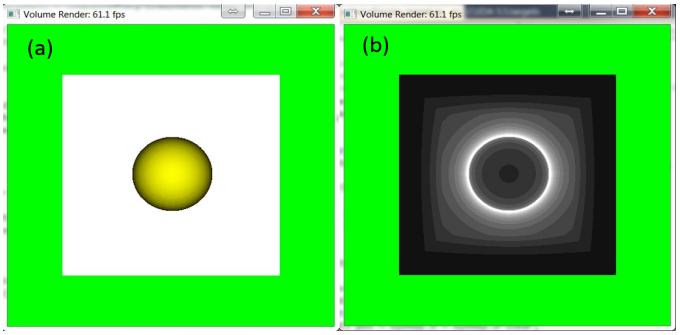

Figure 3.3: 3D signed distance object rendering: (a) ray-casting rendering for the signed distance sphere with radii of 30, shaded by the angle between ray direction and surface normal direction, (b) the same sphere shaded by the scale of actual steps over 15.

It is worthy noticing that in the 3D case, we need to deal with transformation applied on the 3D models. Instead of transform each voxel to a new position after applying the transformation matrix, we choose a more efficient way for rendering. Basically, we apply the inverse matrix on the view point and ray for casting, which is equivalent to transform the whole models in terms of rendering.

3.3 Some Special Rendering Modes: Collision Alert, Cut through view and Skeleton Data Visualization

The first example showed here is to alert users whenever there is collision detected. Collision detection might be difficult in b-rep system since it is struggling to classify a point's membership for different meshed objects in the real time. However, in our voxel modeler, it is straightforward to check whether a voxel is inside an object or not by sampling the distance value from original signed distance object on the global grid. Once we obtain the sampled distance value from object A and B respectively, it is easy to determine whether there is collision by checking whether both sampled value is negative, indicating this voxel sitting inside both of objects. Once there is any voxel detected as collision voxels, the background is changed from gray to red. As Fig. 3.4 shows: (a) shows two disjoint cubes, while (b) renders the background as red, indicating those two cubes have some voxels overlapped inside both of the cubes.

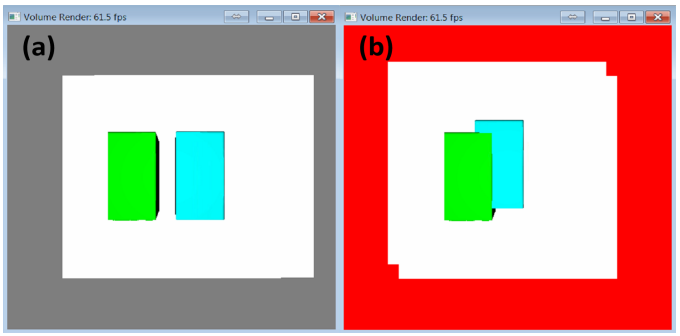

Figure 3.4: collision detection between two SDF-rep cubes: (a) two disjoint cubes without detection, the background is gray (b) two cubes have some part overlapped, thus collision detected, making the background red.

Another interesting viewing mode is the cut through viewing. This mode attracts the users when there is material function applied inside the objects, where you can see the material distribution throughout the whole objects. In this mode, besides taking the ray

which does hit the surface of the objects, we further consider those rays getting inside of the objects, and further hit the virtual cutting plane. Of course, the color or material distribution inside the objects is dependent on our material functions. For example, in Fig. 3.5 (b), there is single material dominating the whole talus, thus only the yellow color can be seen. While, Fig. 3.5 (c) shows a special material function applied: the blue color indicates deeper position inside the objects, while yellow color indicates the position close to the surface. Finally, the Fig. 3.5 (d) shows a coordinates based material distribution, where green material dominates the voxels having large z coordinate value.

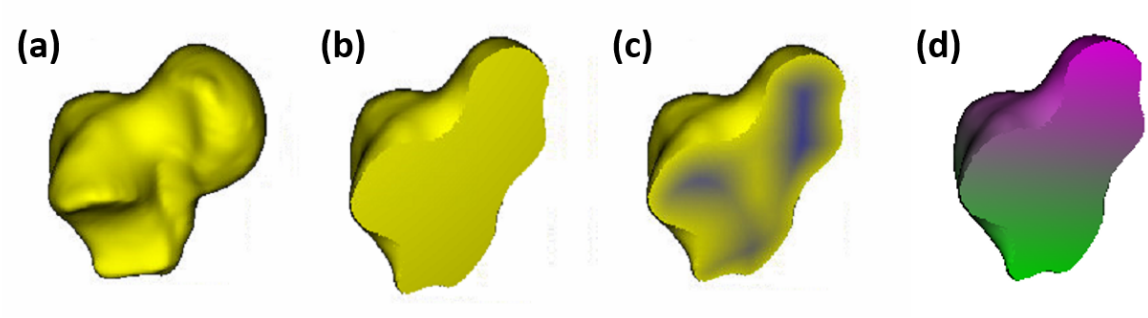

Figure 3.5: apply different material functions on a single SDF model: (a) a SDF-rep talus render at the zero offset of surface, (b) a cut through view of the talus with single material, (c) a cut through view of the talus with the material function: blue color indicates the large distance inside the object, yellow color indicates the small distance inside the object, (d) a cut through view of the talus with the material function: green color indicates the rear side of the bone, red color indicates the front side of the bone (coordinate based).

Last but not least, our voxel system has the capability to render the skeleton data of a SDF-rep model. Basically, all the skeleton data is a grid point whose gradient value is off 1 by certain threshold. Once we obtain those grid points, we assign virtual spheres with radius of 1 on those grid points. Instead of shooting ray to look for the intersection among those spheres, we directly use parametric rays and spheres to describe the problem and get

the analytic solution whether there is any intersection between the rays and spheres, which is much more efficient compared with root finding like before. Fig. 3.6 (a) shows a original SDF-rep hand model, while Fig. 3.6 indicates the blue skeleton data of the hand model, where the shading is dependent on the angle between the ray direction and surface normal direction of spheres.

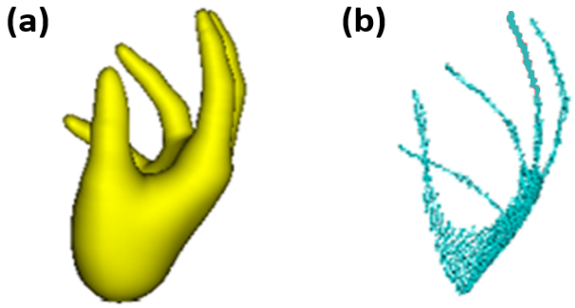

Figure 3.6: skeleton rendering for a SDF-rep hand model: (a) original rendering mode for a SDF-rep hand model, (b) skeleton render mode for the skeleton data of a SDF-rep hand model.

Chapter 4

GEOMETRIC OPERATIONS FOR CREATING, COMBINING, AND EDITING DISCRETE SDF-REP OBJECTS

In this chapter, we present several classes of geometric modeling operations including unary operations (that involve a single SDF-rep input), binary operations (including the usual Boolean operations), sweeping operations. The operations presented include offsetting and sweeping operations that are problematic with traditional b-rep modelers.

4.1 Unary Operation: Offsetting the Surface of Discrete SDF-rep Models

The usual operations of scaling, translating, rotating, and shearing are readily supported in the discrete SDF-rep environment. For rigid body transformations, distances are (by definition) preserved and such operations can be implemented by appending a transformation matrix to the model and applying the transformation to coordinate values to obtain grid coordinates for evaluation of the discrete SDF-rep. Alternatively, the user can choose to compute and store a new grid of signed distance values by applying the transformation to the full grid and storing the signed distance values obtained by interpolating the values in the original grid.

The significant unary operation supported by discrete SDF is offsetting which may alternatively be referred to as thickening and thinning or dilation and erosion. The offset surfaces of the surface of any object are the level sets of its distance function. Thus, SDF inherently include the description of their offsets. Eq. 4.1 describes how the dilation (or thickening) of a solid with SDF-rep f by distance d is achieved by simply adding d to the function value (or to each value on the grid for a discrete SDF-rep). Alternatively, the SDF-rep or stored signed distance grid can be left unchanged, and d can be specified as the new threshold value

in the PMC test.

$$\begin{aligned} &\Omega = \{(x,y,z)|f(x,y,z) < 0\} \qquad (4.1a) \\ &Dil(\Omega, d) = \{(x,y,z)|f(x,y,z) - d < 0\} = \{(x,y,z)|f(x,y,z) < d\} \quad (4.1b) \end{aligned} \tag{4.1}$$

The original toy head, in a box with the side length of 128 as shown in Fig. 4.1(b), is offsetted by minus 5 (grid spacing), making the new signed distance toy head thinner as shown in Fig. 4.1(b). Similarly, in Fig. 4.1(c) shwo, by adding positive 5 to the original signed distance field, a fatter toy head is created in real time straightforward without re-constructing polygonise approximation for meshes objects[38, 39, 40], compared with the polygonal modeler employed in most of commercial software currently.

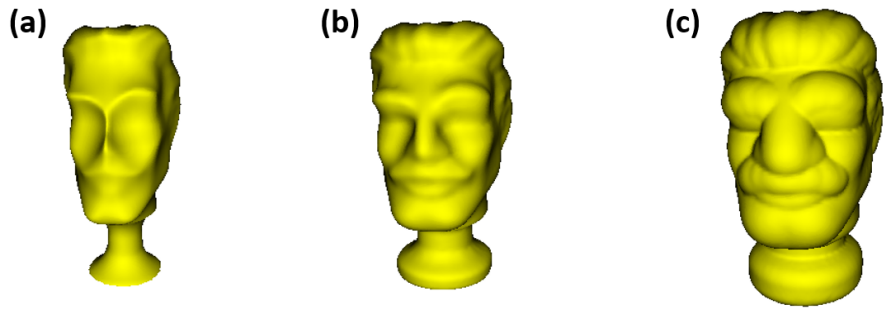

Figure 4.1: Making objects thinner or fatter by adjusting the offset value: (a) discrete signed distance toy head with negative offset, (b) discrete signed distance toy head with original zero offset, (c) discrete signed distance toy head with positive offset.

An interesting application of integrating the boolean operations (to be discussed in the next section) and surface offsetting is to create a shell version of 3D object. From users' perspective, we need to take the absolute value to define a thin surface shell and then offset a shell of the desired thickness. Here, the first step of creating a shell is discussed in detail below. Firstly, it is necessary to access to the original 3D objects and its offset version, either "thinner" or "fatter". Then, what we need to do is to apply the difference operation

on those two objects without referring to complicated CSG tree, in other words, merely applying $max(model_{original}, -model_{offset})$ to get the shell version $model_{shell} = model_{original} - model_{offset}$. For clearer explanation, the procedure of creating shell version of objects is illustrated by the example of signed distance tooth shown in Fig. 4.2. The original signed distance tooth is shown in Fig. 4.2(a) as blue, while the green one in Fig. 4.2(b) indicates the offset tooth, thinner than the original tooth. Fig. 4.2(c) shows a cross section view of the shell version of signed distance tooth after applying the difference operation of $(a) - (b)$. The shell version could be potentially used to make tight container of fragile products or even make customized tightly fit suits, gloves or masks.

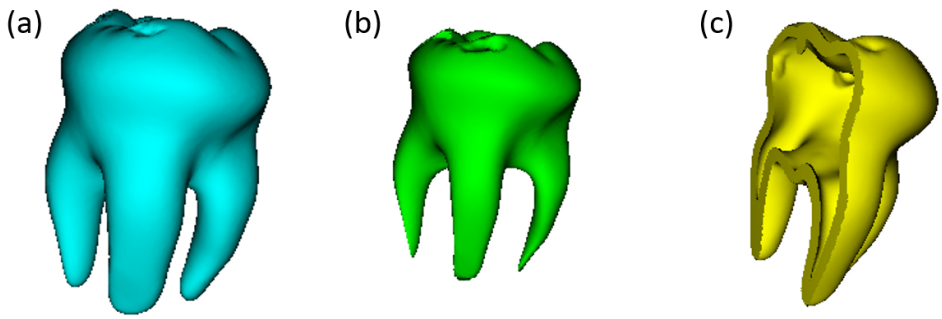

Figure 4.2: Creating shell version of signed distance tooth: (a) the original signed distance tooth, (b) the offset signed distance tooth, (c) the shell version of tooth after the difference operation (a)-(b).

4.2 Binary Operations: Boolean Operation

There are several methods to apply boolean operations on implicit solids defined by f_1 and f_2 in Fig. 4.3[41, 42, 43]. Among them, using Max or Min function is the simplest approach [4]. For instance, the function based union operation $f_1 \cup f_2$ can be realized by $\min(f_1, f_2)$ function. Correspondingly, the Fig. 4.3 shows the new distance f_{dotted} after union operation applied on two original distance functions f_1 and f_2. In Fig. 4.3(a), instead of obtaining the correct output interval for union operation as f_{solid}, the f_{dotted} actually offers

the interval only with correct external part but distorted internal part, waiting to be fixed by upwind differencing method to solve eikonal equation. Eventually, the fast sweeping method recovers the correct signed distance function for two overlapping intervals, presented here as a black solid line or f_{solid}. While, Fig. 4.3(b) and Fig. 4.3(c) show the correctness of f_{dotted} representing the new signed distance field after applying the union operation on the contained intervals and disjoint intervals respectively.

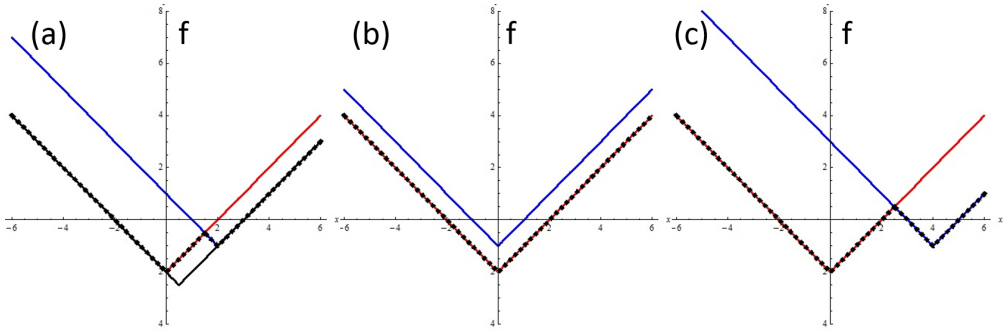

Figure 4.3: Union operation for two signed distance function f_1 (red) and $f_2(blue)$: (a) Min for union of overlapping intervals, (b) Min for union with contained intervals, (c) Min for union of disjoint intervals.

Similarly, the formula representation of intersection operation and difference operation on signed distance solids can be obtained straightforward as Eq. (4.2) below, where, a followed step of applying upwind differencing scheme to solve eikonal equation is used to fix the destroyed signed distance field if needed.

$$\begin{aligned} f_1 \cap f_2 &= \max(f_1, f_2) \\ f_1 \cup f_2 &= \min(f_1, f_2) \\ f_1 - f_2 &= \max(f_1, -f_2) \end{aligned} \tag{4.2}$$

The same conclusions for all situations presented above in 1D scenario still persist when we move to higher dimensional objects. The simple example in Fig. 4.4 provides intuitive sense of applying the boolean operations on the 3D signed distance tooth (converted from

STL mesh model), a bear head (converted from point cloud scanned data) and a sphere (directly defined by a signed distance function). Fig. 4.4(a) is showing the the result after applying union operation on a tooth and a toy head. Fig. 4.4(b) shows the difference result between a bear head mask and a human head model, in other words, the picture is showing the result of bear head model minus the human head model. Fig. 4.4(c) indicates the intersection result between a sphere and the toy head model.

Note that the user may re-position objects prior to specifying a Boolean operation, so the grids of points specifying the input models may not be coincident. The approach implemented in the modeler is to resample the discrete SDF-rep inputs on a common grid where function value comparisons can be readily performed to produce a grid of values to define a discrete f-rep for the output solid. Resampling is performed by invoking the interpolation scheme specified for the inputs and, while we often specify wavelet-based interpolation methods to have control over the smoothness of the interpolation, for resampling operations a multi-linear interpolation often produces satisfactory results more efficiently. Once we re-sample both model A and model B on the common global grid points, or just resample on one of the existing grid for reducing the amount of resampling required. the max and min function can be applied to achieve the boolean operations as discussed above.

For each Boolean operation, Ensz [4] identified regions (specifically the interior or exterior of the resulting solid) where the signed distance property is preserved. In some cases, the signed distance property is preserved globally, and the result is immediately an SDF-rep. In other cases the signed distance is preserved in either the interior or exterior region; the signed distance values in that region are valid and an upwind-differencing computation is applied to complete to compute the signed distance values on the reminder of the grid. Note that there is no Booelan operation that fails to preserve the signed distance property in both the interior and exterior regions. Fig. 4.5 illustrates the union of two discrete SDF-rep cubes. Fig. 4.5 (a) shows two cubes with the same dimensions that are defined directly as discrete sampling of closed-form signed distance functions. Fig. 4.5 (b) shows the cubes positioned in a overlapping configuration, and Fig. 4.5 (c) shows a cut-away view of the

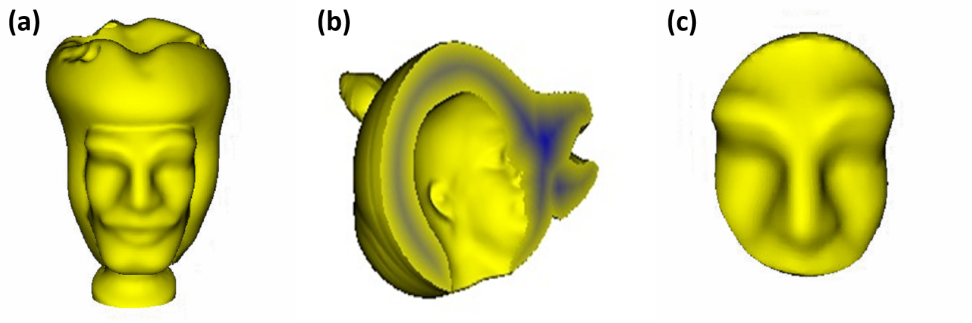

Figure 4.4: 3D signed distance objects bool operation: (a) is showing the the result after applying union operation on tooth and a toy head. (b) shows the difference result between a bear head mask and human head model, in other words, the picture is showing the result of bear head model minus the human head model. (c) indicates the intersection result between a sphere and the toy head model.

interpolation of the grid of function values obtained by taking the minimum of the input values; i.e. $\min(f_1, f_2)$. While $\min(f_1, f_2)$ correctly defines the geometry of the union, there is a bright region in the center of Fig. 4.5 (b) where vertical displacement, which changes the distance to the surface does not change the function value. Thus $\min(f_1, f_2)$ does not define a signed distance function throughout the interior of the union, However, $\min(f_1, f_2)$ does define a signed distance function on the exterior of the union. The discrete SDF-rep for the union of cubes is obtained as the valid exterior values together with interior values computed by upwind differencing, and a cut-away view is shown in Fig. 4.5 (d).

4.2.1 Example Application: Design of Lattice Structures for 3D Printing

To illustrate the robustness of Boolean operations on discrete SDF-rep models, we consider the design of lattice structures for 3D printing. In particular, we consider tetrahedral lattices which can be thought of as the dilation of the edges of tetrahedral meshes such as those produced by software packages such as Tetgen [44] that are used to create tetrahedral meshes of polyhedral. An example of a lattice structure, corresponding to the tetrahedral mesh generated for polyhedral teddy bear, is shown in Fig. 4.6. In a polygonal b-rep modeler, such

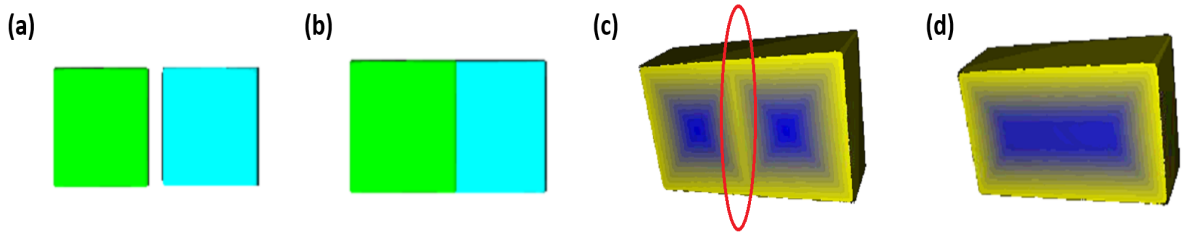

Figure 4.5: upwind differencing repairs the signed distance field after union operation: (a) two cubes with the same dimensions are defined directly by the signed distance function, (b) push one cube toward the other until there is collision detected, (c) apply the union operation on those two cubes without using upwind differencing to repair the distorted distance field, (d) apply the union operation on those two cubes, using upwind differencing to repair the distorted distance field.

a lattice structure would be created as the union of large numbers of polygonized cylinders and, if pairs of cylinders are nearly co-axial, computations associated with intersecting and trimming the bounding polygons become numerically sensitive. In fact, several efforts to construct such models with commercial CAD packages were unsuccessful.

4.3 Modeling SDF-rep Solids with Combination of Boolean Operations

Sweeping operations involve operations on the classification of points with respect to a solid undergoing a continuous family of kinematic transformations. Typically the continuous family of kinematic transformations is approximated by a discrete sequence of near-identity rigid body transformations and, as in the case of lattice structures considered above, b-rep modelers tend to run into difficulties involving numerical sensitivity of computing intersections of nearly parallel boundary entities [Insert references]. To compute swept discrete SDF-rep solids, we also approximate the kinematics as a discrete sequence of near-identity rigid body transformations, but we avoid the problems associated with computing boundary intersec-

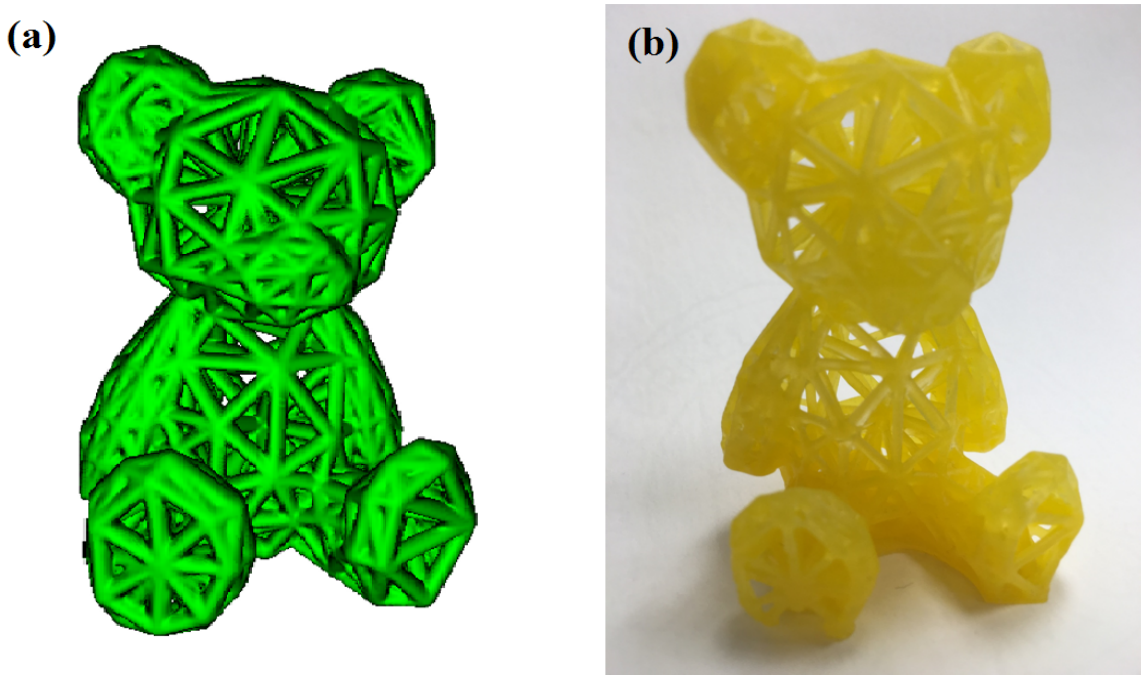

Figure 4.6: tetrahedron bear: (a) tetrahedron bear in our SDF-rep CAD system, (b) printed tetrahedron bear by vat polymerization.

tions. A sweep operation is accomplished as a sequence of Boolean operations, and the parallel implementation of function evaluations, value comparisons, and rendering results in a formulation that supports real-time interactivity. Note that a discrete f-rep suffices during the interactive sweep, and completion of the grid to re-establish the signed distance property globally only needs to be done once at the end of the sweep construction.

4.3.1 Sweep Cut or Sculpturing

Sculpturing or sweeping cut is an interesting feature for designers since it mimics material removal processes such as milling, drilling, and turning during which the material removed from a piece of stock corresponds to its intersection with a tool of fixed geometry. In Fig. 4.7, I present an example in which a discrete SDF-rep tool is used to carve a virtual model that is then fabricated by 3D printing.

Fig. 4.7(a) shows that the spherical tool being used to carve the letter "U" into a plate. Fig. 4.7 (b) - (d) show the letters "W", "D", and "Z" being carved into the plate using spherical cutting tools of varying size. Sculpturing or sweep-cutting in the discrete SDF-rep modeler is straightforward and supports a variety of tools and tool sizes. The procedure is realized in real time at the frequency of 67 frames per second with Tesla K20 GPU for parallel computing and GeForce GT 620 for displaying. The physical object resulting from 3D printing the sweep-cut model is shown in Fig. 4.8.

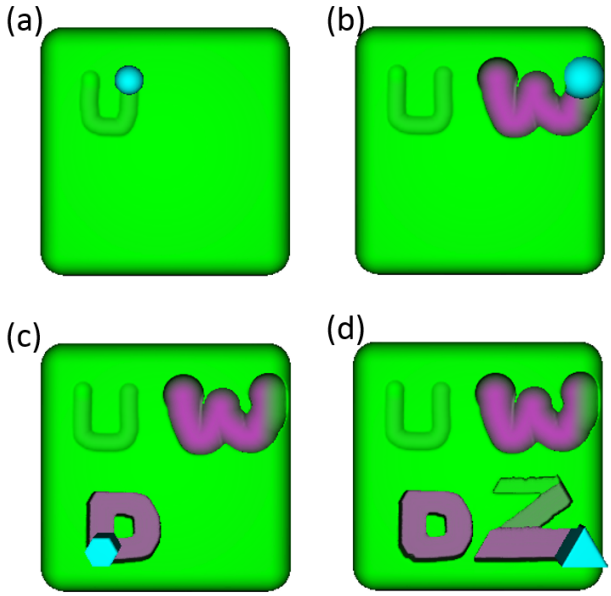

Figure 4.7: Sculpturing patterns on a virtual plate: (a) a sphere chisel is used to carve an U, (b) a larger sphere chisel is used to carve an U, (c) a hexagon chisel is used to carve an U, (d) a triangle chisel is used to carve an U.

4.3.2 Sweeping Construction

Just as a sweep-cut was realized as a sequence of difference operations in the previous section, we now present an example that constructs a swept solid as a sequence of union operations.

Figure 4.8: Real 3D printed sculptured plate with designed pattern.

Again, a discrete f-rep suffices during the interactive construction and completion of the grid by upwind differencing is performed once after the sweeping transformations are done.

Fig. 4.9(a) shows, a discrete SDF-rep model of a talus, a bone in the ankle, derived from segmented CT scanner data. In this example, we apply a sequence of small transformations corresponding to the talus revolving along a circular path while simultaneously doing one full rotation about the direction of motion. Fig. 4.9(b) and Fig. 4.9(c) show the talus sweeping through the first quarter and the first three quarters of motion respectively. The continuous sweeping motion is approximated as a sequence of 300 discrete rigid body transformations which suffies to produce a model without apparent artifacts as shown in Fig. 4.9(d).

Fig. 4.10 (a) shows four regulary spaced key frames of the sweep, and Fig. 4.10 (b) shows an exploded view of the key frame tali along with the partial swept solids that connect them to form the complete swept solid shown in Fig. 4.10 (c). Fig. 4.11 (a) and (b) show the corresponding physical objects fabricated by 3D printing on a powder-bed binding system, and discretization artifacts cannot be readily detected by either visual or tactile inspection.

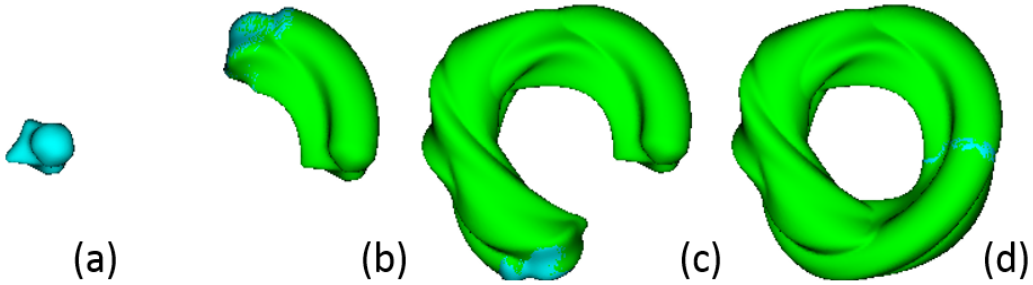

Figure 4.9: Sweeping mobius strip for talus: (a) a single talus bone waiting for sweeping construction, (b) the first quarter of the talus mobius strip, (c) three quarters of the talus mobius strip, (d) fully constructed talus mobius strip.

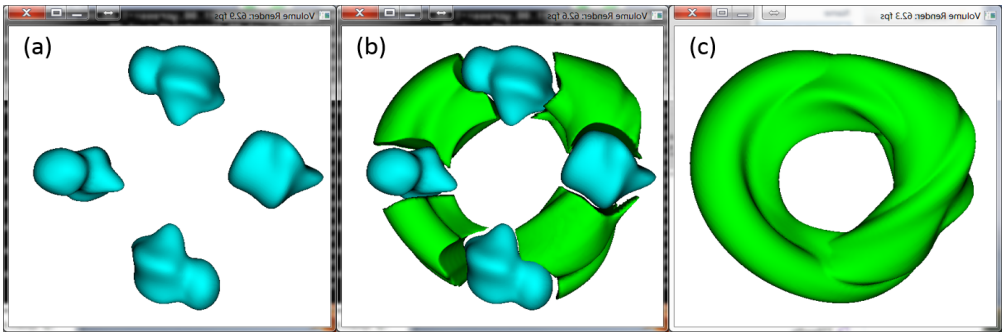

Figure 4.10: Modeling swept volumetrically-digitized solids: (a) four copies of volumetrically digitized talus as key frames in a double rotational sweep, (b) exploded view of key positions and interstitial swept solids, (c) digital model of swep digitized solids

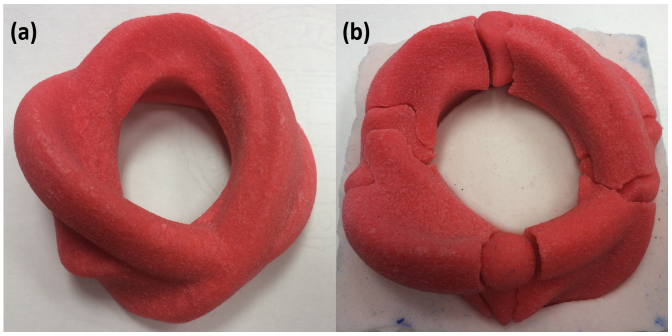

Figure 4.11: Real 3D printed swept talus: (a) fully swept version of talus by applying union operation on a single SDF talue model with 300 different configurations, (b) a explode figure of the swept SDF talus with four key positions and four separate swept sections.

Chapter 5

DEFORMING SDF-REP MODELS BY DISCRETE GEOMETRIC SKELETONS

While skeletal editing tools have been proposed for b-rep models [45], significant efforts by numerous researchers have yet to provide robust skeletonization of b-reps [46]. Thus the current state of affairs is that skeletal editing methods would support useful modeling operations including dilation, erosion, and thickness determination, but such operations are not generally available in current CAD systems. In this chapter, we demonstrate that discrete SDF-rep models support a discrete version of the geometric skeleton that can be efficiently computed and readily edited. Moreover, we show that the upwind differencing scheme (used earlier for completing SDF grids for import of digitized objects or re-establishing the signed distance property after Boolean operations) can be used to compute the SDF grid from the skeletal values to effectively "reflesh" a solid after skeletal editing.

Our method for identifying skeletal grid points exploits the fact that a signed distance function satisfies the eikonal equation $|\nabla f| = 1$ wherever the gradient exists. The singular points of the SDF correspond to the skeleton, and we obtain a discrete approximation by computing the components of the gradient by applying a finite difference or wavelet connection coefficient vector as a stencil on the SDF grid and then selecting points where the magnitude of the gradient differs from unity by more than a specified threshold value. The skeletal data consists of the skeletal points along with a radius for the associated (approximately) maximal sphere which is given simply by the magnitude of the SDF at the skeletal point. The sections below present two applications of skeletal editing of signed distance solids, creating a custom-fit mask and re-configuring a model of a human hand.

5.1 Manipulating Skeletal data and "Refleshing" SDF-rep Models

In this section, two application examples are presented: one involves designing customized masks or helmets, the other focuses on the human part 3D model deformation. Both examples are implemented by transforming the internal skeleton to a new configuration and refleshing the full discrete SDF-rep by reconstructing the signed distance field. In both cases, a 2D version is presented for simplicity of exposition followed by a full 3D version.

5.1.1 Customized helmets or masks

A bear head contour is prepared for further editing as shown in Fig. 5.1(a). As discussed above, the geometric discrete skeleton is extracted wherever the gradient of distance field at certain grid point is off 1 by the manually set threshold. Fig. 5.1(b) illustrates the internal distance field with brighter shading indicating great internal distance. Note that the shading value increases moving inward from the wall until a central bright region is reached. Beyond that point, a new nearest surface point becomes relevant and the shading value again decreases. The central brightest region corresponds to the skeleton. Fig. 5.1 (c) shows the skeletal points identified as described above: compute the gradient of the SDF by correlation with a central difference or wavelet connection coefficient stencil, and select points where the magnitude of the gradient differs from unity by more than a threshold value. Fig. 5.1 (d) illustrates a simple case of skeletal editing where the portion of the skeleton in the mouth region is selected and re-positioned, but each skeletal point retains its radius value. Fig. 5.1 (e) is shaded according to distance to illustrate the SDF obtained by employing an upwind differencing scheme to construct the discrete SDF-rep of the edited bear head. The refleshed version of the edited head is shown in Fig. 5.1 (f).

We now consider the design of a face-fitting mask or helmet to demonstrate that the skeletal editing method described above extends directly to 3D. We start by importing a 3D bear's head model, shown in Fig. 5.2(a), into the discrete SDF-rep modeler. The skeletal grid points of the model are determined as above except for the need to compute 3 components

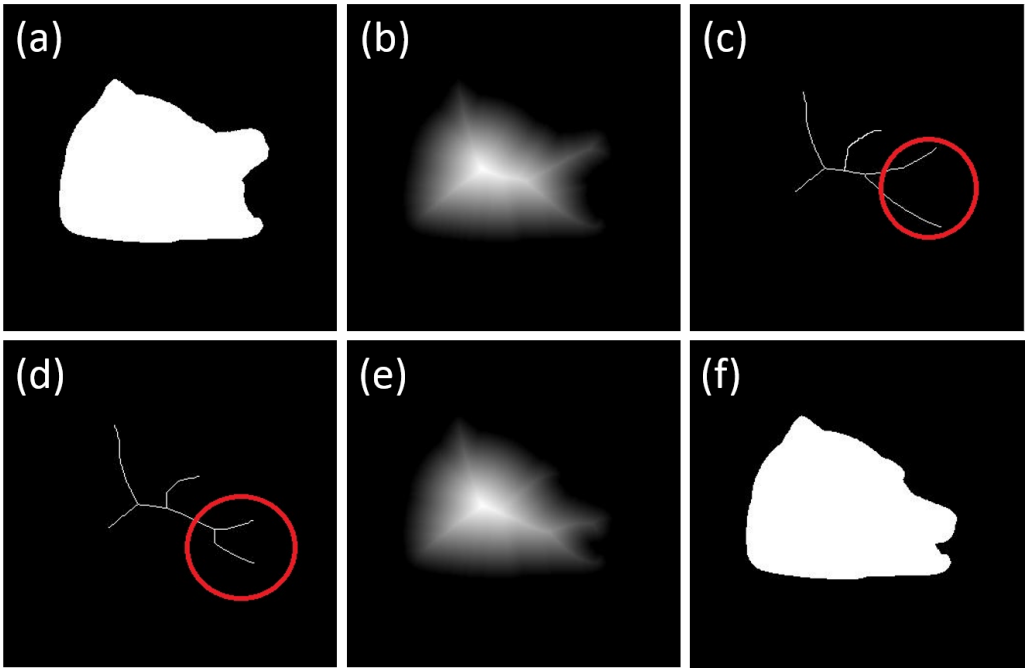

Figure 5.1: 2D signed distance object manipulation: (a) original 2D signed distance bear head contour, (b) original magnitude of distance field of the head contour, (c) extracted geometric discrete skeleton of original bear head contour, (d) edited geometric discrete skeleton, (e) "refleshed" gradient field of signed distance space based on the edited skeleton, (f) "refleshed" bear head contour.

of the gradient. The skeletal points are illustrated in Fig. 5.2(b), and the bear head model and its skeletal points are shown together in Fig. 5.2(c). Once again, the skeletal editing involves selecting the skeletal points of the mouth and repositioning them as shown in Fig. 5.2 (d), which shows the head with its skeletal points. After the editing of original geometric discrete skeleton, the new skeletal configuration is shown in Fig. 5.2(e). Eventually, the deformed signed distance bear head masks is "refleshed" by the upwind differencing scheme in Fig. 5.2(f).

The example demonstrated above shows the convenience of deforming customized objects by editing the discrete geometric skeletons. This approach provides a real time interaction between users and software, which simplifies the 3D signed distance model deforming through

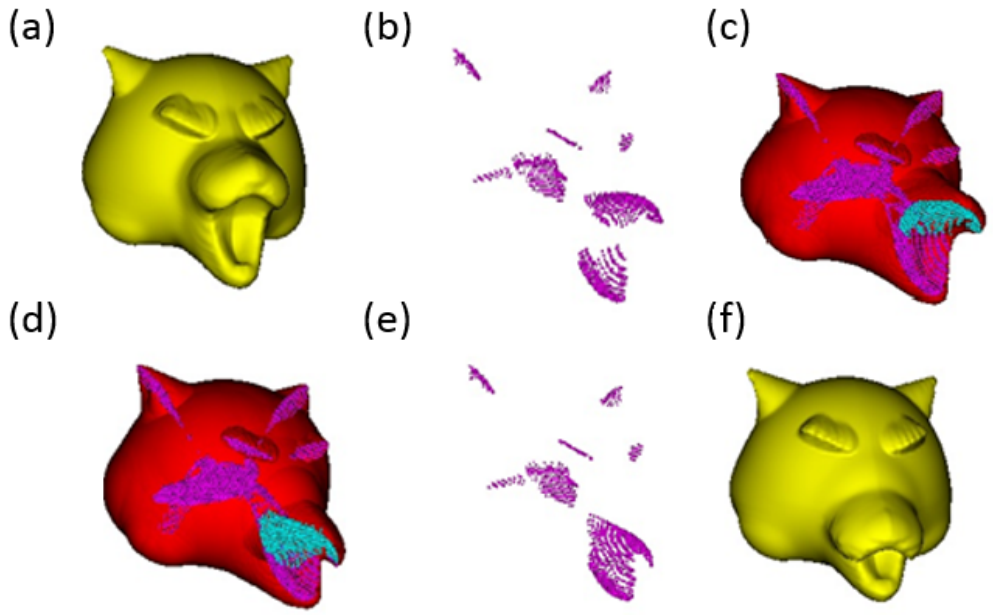

Figure 5.2: 3D signed distance object manipulation: (a) original signed distance bear head mask, (b) original geometric discrete skeletal of the original bear mask, (c) chosen skeleton data (blue) and the rest skeleton data (purple) of the original bear head mask (red), (d) edited chosen skeleton data (blue) and the rest skeleton data (purple) of the edited bear head mask (red), (e) edited geometric discrete skeleton data, (f) deformed bear head mask reconstructed from those edited skeletal.

lower dimensional skeletal points editing.

5.1.2 Reconfiguring A Model of The Human Hand

Here, we present a 3D human hand model which can be customized by easily editing its discrete geometric skeleton. To illustrate the idea clearly, we start with a 2D example of deforming a finger, where the original finger model is shown in Fig. 5.3(a). Extracting discrete geometric skeleton is achieved by computing the distance gradient field, and the magnitude of distance value is shown in Fig. 5.3(b), where, the bright central region contains the skeleton. The skeletal grid points are again identified as points where the magnitude of

the gradient of the signed distance differs from unity by more than a threshold value. The discrete skeleton points are visualized in Fig. 5.3(c), that also shows a red dot labeled Pivot A which the user has designated as a pivot point or joint of the finger. To edit the skeleton, the user selects a set of skeletal points (in this case, the points in the region from the chosen joint position to the skeletal "fingertip") and applies a transformation (here a rotation about the axis normal to the page through the pivot) to relocate the selected points. Any transformed skeletal points, which do not lie on the grid, are rounded to the closest adjacent grid points. Eventually, the signed distance field is reconstructed or "refleshed". The internal signed distance field computed from the edited skeletal data is shown in Fig. 5.3(e), and the edited finger model is shown in Fig. 5.3(f). Note that performing such an operation by applying a transformation to boundary patches of a b-rep model would require dealing with gaps and self-intersections that would need to be trimmed and stitched, such issues do not arise here due to the implicit nature of the modeling representation.

Now we move on to consider a 3D model of a partial hand (distal palm with 4 fingers) to show that the skeletal editing approach described above extends directly to 3D. The only significant difference is that a full 3D specification is required for the rotation axis. (In 2D, specification of a joint location fully specifies the ration axis which must pass through that point and be normal to the plane.) In this example, we also demonstrate editing the full skeletal data; i.e. not only relocating skeletal points, but also changing the signed distance values (and associated sphere radii) at selected skeletal points. The original 3D signed distance hand is shown in Fig. 5.4(a). After computing the gradient of the whole signed distance field, the points identified as skeletal points are shown in Fig. 5.4(b), where, the red dots indicate the set of skeletal points the use has selected for relocation. The edited skeletal points are shown in Fig. 5.4(c), and the edited hand model obtained by re-fleshing the edited skeletal data is shown in Fig. 5.4(d).

An interesting and desired capability of our voxel modeler system is automatically detecting the joints. The challenge is to detect the knuckle joints, and more importantly, the pivot direction, about which, the finger rotate. The method described below exploits the

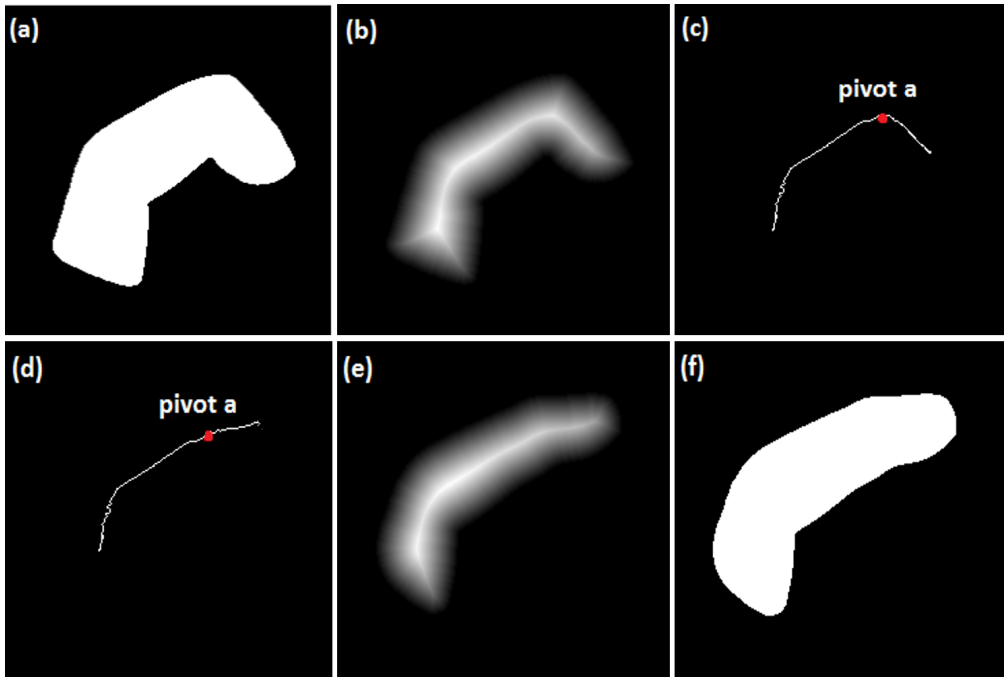

Figure 5.3: 2D signed distance object finger manipulation: (a) original 2D signed distance finger contour, (b) original magnitude of distance field of the finger contour, (c) extracted geometric discrete skeleton of original finger contour, (d) edited geometric discrete skeleton, (e) "refleshed" gradient field of signed distance space based on the edited skeleton, (f) "refleshed" finger contour.

exoskeleton data (skeletal data at points external to the model) to achieve computer-aided identification of joint locations and axes of rotation for models imported in a configuration in which the joints of interest are flexed.

Fig. 5.5(a) shows the hand model in its original configuration as imported into the discrete SDF-rep modeler. Fig. 5.5(b) shows the skeletal data points including the internal skeletal points (shown in blue) and the external skeletal points which are segmented into groups of neighboring points (shown in green and red). Based on the visual appearance, we will refer to these groups of exoskeletal points as "sheets". Note that in addition to the green exoskeletal sheets between adjacent fingers, there is also a red exoskeletal sheet associated with each bent knuckle in the fingers. The system identifies the interior skeletal

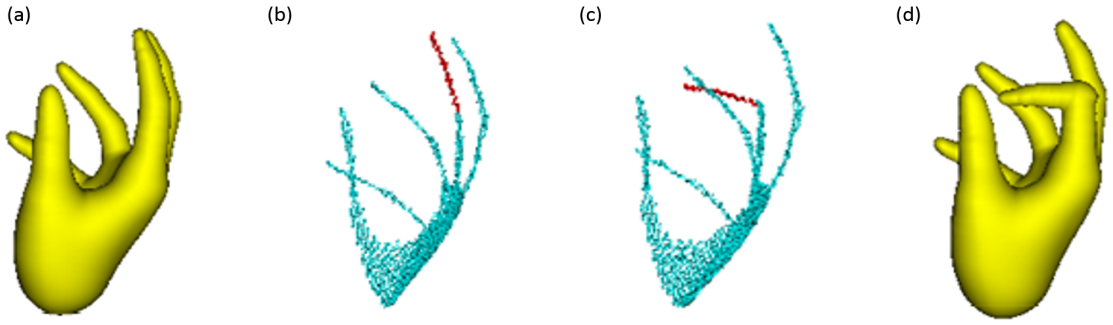

Figure 5.4: 3D signed distance object hand manipulation: (a) original signed distance hand, (b) internal discrete skeleton, blue one is internal skeleton, red one indicates for selected partial skeleton needs to be manipulated later, (c) bent internal discrete skeleton, the red parts of the finger is bent to right angle compared with the one in (c), (d) reconstructed 3D signed distance hand model with bent finger.

point closest to each exoskeletal sheet as a knuckle joint location. Each knuckle in the fingers is treated as a revolute joint, and the system automatically determines the rotation axis by computing the cross product of the following 2 vectors: (1) the normal vector to a best-fit plane through the points in the exoskeletal sheet, and (2) the displacement vector from the joint location to the estimated centroid of the exoskeletal sheet. Fig. 5.5(b) illustrates the situation after the user has selected an exoskeletal sheet (associated with the proximal joint of the third finger from the left) to obtain the joint location and axis and then selected the portion of the finger skeleton from that joint to the skeletal fingertip. The skeletal data, the spheres centered at the skeletal points with radii corresonding to distance magnitudes, is illustrated with Fig. 5.5(c) with the selected skeletal spheres shown in red. Fig. 5.5(d) illustrates the edited skeletal data after the selected portion of the skeleton has been rotated about the aoutomatically identified joint rotation axis, and Fig. 5.5(e) shows the re-fleshed model of the edited hand. Note that, as in 2D, nothing special needs to be done to deal with "self intersections" associated with the editing operation; the discrete SDF-rep automatically retains validity as a solid model. Finally, Fig. 5.5(f) shows the edited model obtained after

re-fleshing from edited skeletal data that includes increasing the radius of the skeletal spheres to dilate the selected portion of the finger.

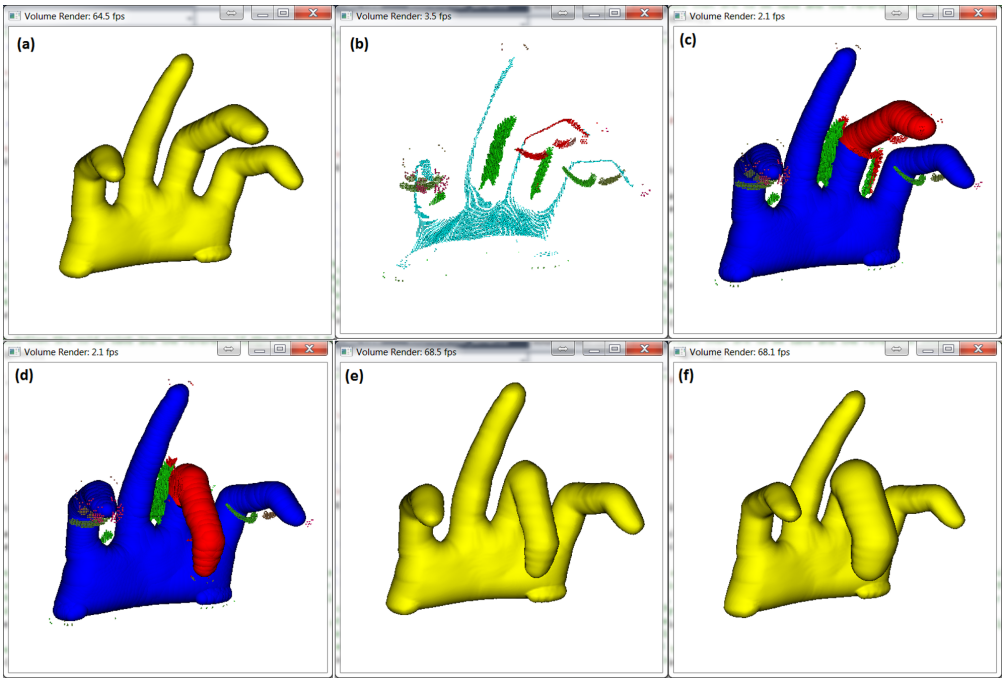

Figure 5.5: 3D signed distance object hand manipulation: (a) original signed distance hand, (b) internal discrete skeleton, blue one is internal skeleton, red one indicates for selected joint by making use of the external skeleton and predict the position of the joint as well as the pivot direction, (c) selected part of middle finger rendered as red and the rest rendered as blue, (d) rotated middle finger about the selected joint, (e) reconstructed signed distance hand without modifying local radius of middle finger, (f) reconstructed signed distance hand with modified local radius of middle finger.

Chapter 6

INHOMOGENEOUS SOLIDS AND ADDITIVE MANUFACTURING WITH GRADED PROPERTIES

As mentioned in the introduction, one of the primary motivations for creating a discrete SDF-rep modeling system is to fully support the new capabilities provided by developments in additive manufacturing/3D printing. In previous chapters we established that discrete SDF-rep models provide a unified format for workflow from digitization by scanning (volumetric or surface) to editing, and now we are ready to address issues associated with fabrication.

The first thing to note is that we can, at any point in the process, invoke marching cubes to convert a discrete SDF-rep model back to a surface polygonization (e.g. an STL file) that would provide a standard input format for that vast majority of additive manufacturing systems. However, we want to pay special attention to AM systems based on powder-bed binding and vat polymerization where the fabrication system literally prints or projects a sequence of images to build a physical part. By applying an occupancy function (which returns 1 at interior voxels and 0 at exterior voxels) to the discrete SDF-rep, we produce a stack of binary (black and white) images suitable for fabricating the desired geometry. (Note that it may be desirable to rescale the image stack/voxel set to match the printer resolution and layer thickness, but such image resampling is a straightforward and readily parallelizable exercise in interpolating the grid values.)

Now we come to the question of describing and fabricating inhomogeneous objects whose composition, properties, or process parameters vary within the interior of the object. Phrases used to describe such objects include graded, spatially varying, and locally controlled properties. The number of descriptions that have been created indicates the desire for such fabrication capabilities; however, the availability of fabrication systems to perform such fabrication

tasks is extremely limited, and that may be associated with the lack of modelers to describe graded objects. That said, there are ways in which some existing AM systems can be hacked to produce graded properties. While powder bed binding systems typically require polygonized b-rep input in STL format, a script that prints a sequence of test images has been used for direct printing of image stacks enabling fabrication of objects digitized from CT scans that are free of polygonization artifacts.

Here we focus on vat polymerization systems. Again, the presumed input format is polygonized b-rep in STL format which serves as input to slicing software that produces the stack of images to be projected into the polymer vat. Following slicing of a dummy STL file, the images produced by the slicing software can be replaced with a stack of images from the discrete SDF-rep modeler to produce parts without having to convert the desired part to STL format. With this capability in hand, we are ready to proceed to creation of graded discrete SDF-rep models.

6.1 Defining Multi-Material Function on Graded SDF-rep Models

The geometry of a discrete SDF-rep object is described by a set of signed distance values on a grid (along with a means interpolating when necessary to determine values between grid points). The approach to describing auxiliary properties is straightforward: simply append an auxiliary grid of values (or an auxiliary function that can be evaluated on the grid points) to specify the property value. Wherever the value of the signed distance (that defines the geometry) is negative (indicating an interior voxel), the auxiliary function value at the corresponding grid point specifies the property of interest. Fig. 6.1 shows a collection of graded discrete SDF-rep objects designed for a scenario in which the lighter shading indicates higher modulus but more costly material compared to the material with darker shading. Fig. 6.1 (c) shows an inhomogeneous I-beam with higher modulus material at the flanges to optimize moment of inertia about the neutral axis (and resistance to a vertical bending load). Fig. 6.1 (d) shows an inhomogeneous pipe with higher modulus material at the outer edge to optimize polar moment of inertia (and resistance to a torsional load). Fig.

6.1 (a) shows a model of a tooth with modulus decreasing with distance from the surface.

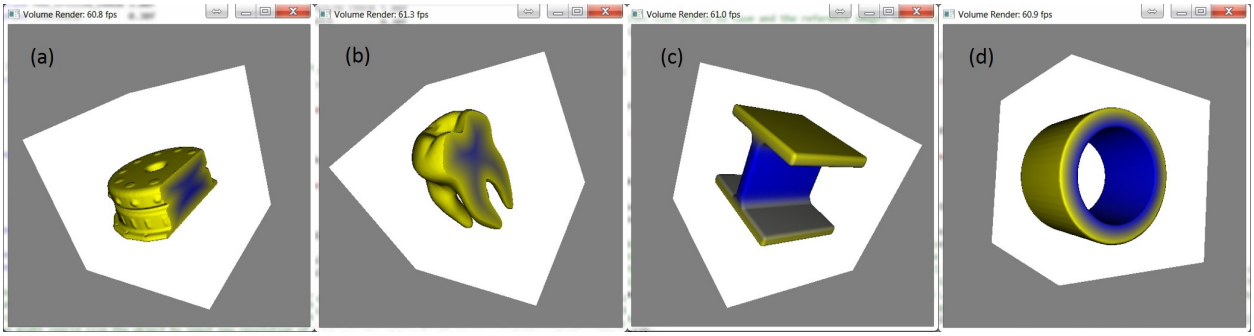

Figure 6.1: (a) chocolate-cream cake with chocolate dominates on the surface and cream dominates inside the object, (b) cost effective two materials tooth with harder, expensive material close to surface and soft, cheap material inside, (c) cost effective two materials i-beam with harder, expensive material on top and bottom plates and soft, cheap material for middle connecting plate, (d) cost effective two materials pipe with harder, expensive material close to outer surface, cheap material close to inner surface.

6.2 Graded Boolean Operations

In Chapter 4 we described how discrete SDF-rep geometry is combined via Boolean operations that compare local values of the signed distance for the operand objects. Now we arrive at the question of how to combine the auxiliary function values when objects are combined. The auxiliary function provides an additional degree of freedom that supports creation of new families of binary operations [47] on graded discrete SDF. Here we present a few of the possibilities choices.

A first approach can be thought of in terms of inheriting properties from the input operand objects. The geometry of the object is determined according to operations described in earlier chapters, and the property value at each point in the operand is determined as a function of the property values at the corresponding point in the operand objects. For example, a binary

operation can be defined so that points in the output object that belong to a single input object inherit the property value from that input object, while points in the intersection of the input objects are assigned a property value that is a function of the property values of the two input property values of the input objects at that point. One possibility is to declare one input to be dominant, so that the output object inherits the property value from the dominant input in the intersection region. Alternatively, the output property value in the intersection region may be some blended function of the property values of the input objects at the point.

Fig. 6.2 shows a configuration of a pair of discrete SDF-rep objects with an auxiliary property: a dark (blue) sphere and a light (yellow) block. Fig. 6.2 (b)-(d) show cut-away views of the inhomogeneous discrete SDF-reps resulting when the property values in the intersection region are specified by (b) the value from the dominant block, (c) the value from the dominant sphere, and (d) interpolating between the input property values with a weighting derived from the distance values.

Note that new property blending operations can be defined by combining any of the previous geometric operations with a specification for output property values specified as a function of input property values, spatial coordinates, or other desired parameters. For example, we used sequences of near-identity Boolean operations to define the geometry of a swept solid, so we can apply a blended union of inhomogeneous discrete SDF-rep objects to define an inhomogeneous discrete SDF-rep swept solid as illustrated in Fig. 6.3. Fig. 6.3 (a) shows an inhomogeneous version of the talus (the same one that was used to create a swept homogeneous discrete SDF-rep solid in Chapter 4) with property value varying linearly along a coordinate axis. The sequence of transformations is again applied, but now with a union that blends property values with the talus in its most recent configuration as the dominant input. Partial sweeps are shown in Fig. 6.3 (b) and (c), and the full-circle inhomogeous discrete SDF-rep object is shown in Fig. 6.3 (d).

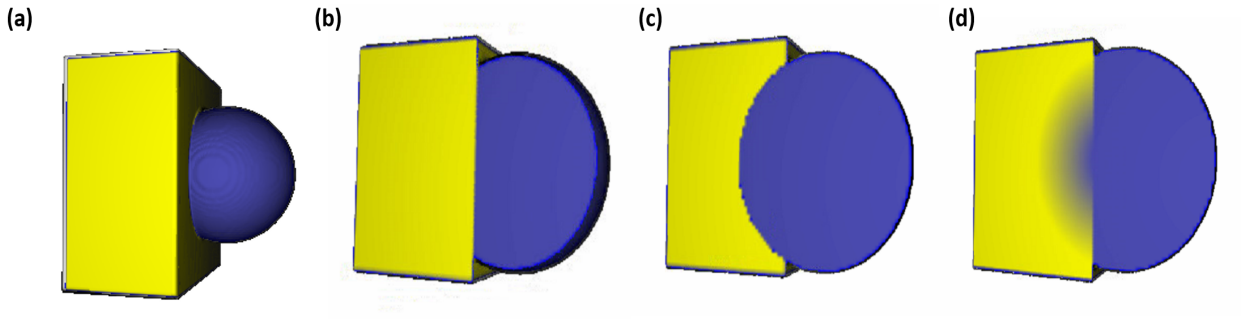

Figure 6.2: Apply union operation on objects with different material property: (a) apply union operation between one cube with material A (yellow) and one sphere with material B (blue), (b) a cut through view of union result: using material A (yellow) to dominate those overlapped voxels, (c) a cut through view of union result: using material B (blue) to dominate those overlapped voxels, (d) a cut through view of union result: using distance based blending function between material A (yellow) and material B (blue) for those overlapped voxels.

6.3 Designing Multi-material Human Nose

3D printing is not a new concept, but until recently, it is rarely seen the reports about printing any human centric part like organs, which brings lots of interesting possibilities and broadens the applications in this field. Not long ago, researchers in Princeton University and Johns Hopkins University explored 3D printing human ears via 3D printing a cell seeded hydrogel matrix in the anatomic geometry of human ears[48], biopolymer tissue scaffolds[49, 50, 51], multi-polymer microstereolithography for hybrid opto-MEMS[52], where, they achieved 3D interweave biological tissue for enhancing the functionality of matrices tissue. The model of the ear can be obtained by 3D reconstruction from scanning a real person's ear or directly drawing an ear in CAD system. Here, in our example, we present a 3D human nose model which can be customized by editing its discrete geometric skeleton. This approach is worth considering when someone who lost the nose due to injury or other accidents, thus, needs

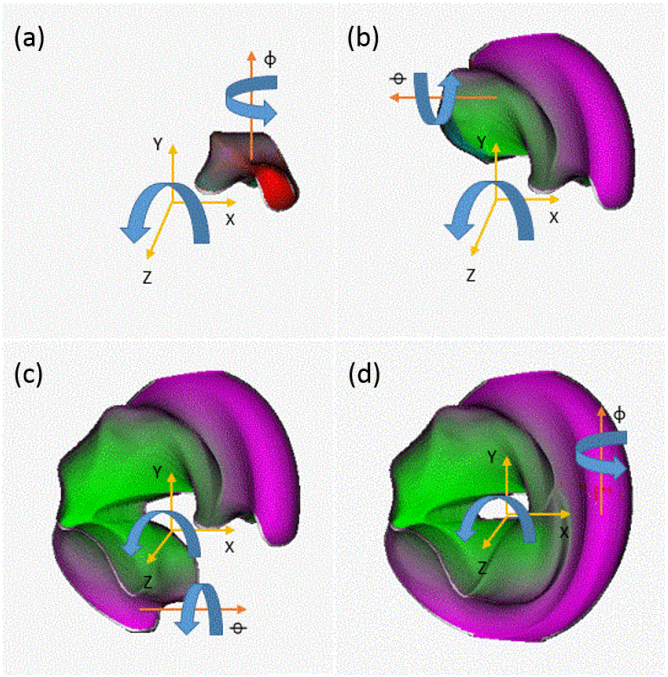

Figure 6.3: Graded swept structure: (a) a single graded signed distance bone, (b) the first quarter of sweeping construction, (c) the first two quarters of sweeping construction, (d) the whole graded swept signed distance bone structure.

a new nose which can be reshaped based on the requirements from users. The generic nose model is shown in Fig.6.4(a). Extracting discrete geometric skeleton is followed by computing and capturing singularity points in this signed distance field, and the skeleton points are visualized in Fig.6.4(b). For specifically editing certain skeleton points, we need to select those skeleton points which are rendered as blue in Fig.6.4(b). Having chosen the potential discrete geometric skeleton points need to be edited, we can edit the skeleton points by transforming, and in our case, we aim to making the new nose high-arched shown in Fig.6.4(d)., compared with original one shown in Fig.6.4(a).

Now, let's move on to the key procedure of this example, which is to create inner porous scaffold nose with thin shell. The scaffold structure in tissue engineering is desirable for cell culture in vitro. Therefore, firstly, a scaffold structure needs to be defined as shown in

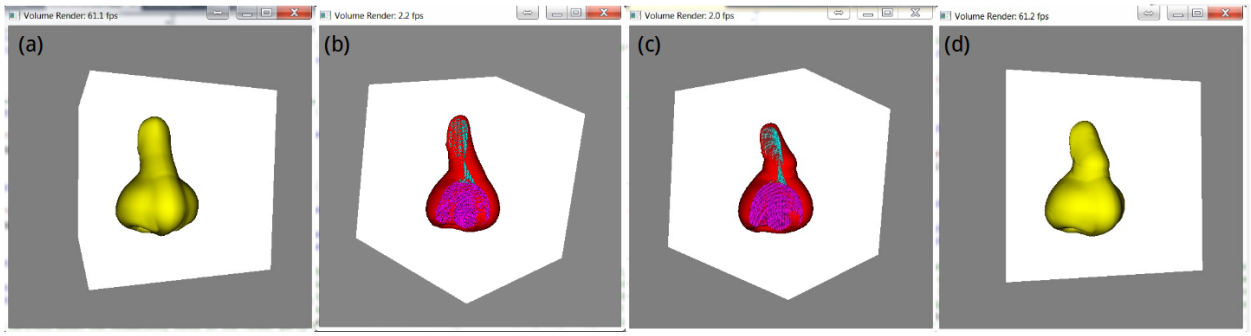

Figure 6.4: (a) original signed distance nose, (b) original discrete geometric skeletal of the original nose, (c) edited discrete geometric skeletal of nose, (d) high-arched nose reconstructed from the edited skeletal points.

Fig.6.5(a). The scallfold structure is downloaded from "Thingiverse" as mesh file and then converted to SDF-rep model. Then, a negative offset nose("thinner version of the original nose") in Fig.6.5(b) is used with scaffold structure for the intersection operation, shown in Fig.6.5(c). Therefore, a scaffold negative offset nose is created in Fig.6.5(d). Fig.6.5(e) shows the inverted scaffold nose which is subtracting the scaffold negative offset nose in Fig.6.5(d) from the original negative offset nose Fig.6.5(b). Eventually, the inner porous structure nose with thin shell derives by subtracting the inverted scaffold nose in Fig.6.5(e) from the original non-offset high-arched nose Fig.6.5(d). With this method, the inner scaffold structure is preserved as the same as previous scaffold negative offset nose, and combined with thin shell as shown in Fig.6.5(f). Besides, our graded material distribution, shown in Fig.6.5(f) as blue and yellow, is described to design the nose, which is motivated from tissue engineering that the material of the core of scaffold network can be stiffer (indicated as blue) to support the whole structure, while the other softer material (indicated as yellow) is used to cover on the surface of core structure [48].

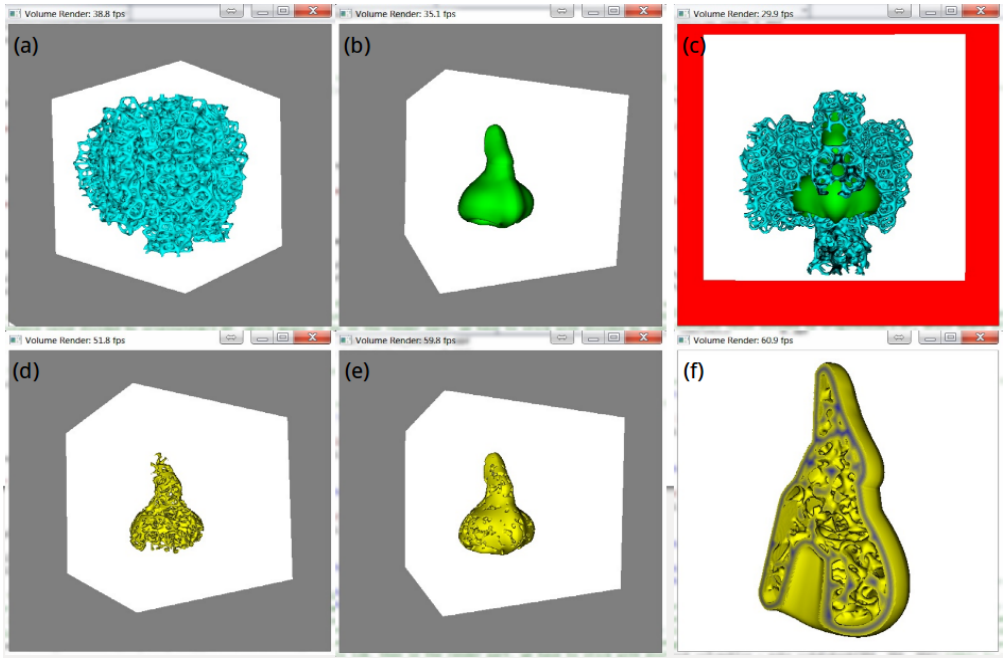

Figure 6.5: (a) scaffold structure, (b) negative offset nose, (c) intersection operation on scaffold structure and the negative offset nose, (d) scaffold negative offset nose, (e) inverted scaffold nose, (f) final nose model with inner porous structure and thin shell.

6.4 Truss Structures with Graded Material Properties

In this section, we describe the results of using a vat polymerization AM system to fabricate objects designed as inhomogeneous discrete SDF-reps. The vat poymerization system uses light projected from a digital light projection (DLP) device to build objects by curing photopolymer contained in a vat. By projecting gray scale images, we achieved control of spatial variation of light intensity, degree of polymerization, and elastic modulus of the cured material. Interactions with colleagues who are interested in the properties of materials with periodic substructures led us to choose graded versions of the unit cell of an octet truss as the desired output.

The pipeline of using our implicit function description for different geometries combined

with material function applied on each voxel is illustrated in Fig. 6.6. First, the continuous signed distance function description for a capsule is presented in the system, as the pure yellow capped cylinder shown. Then, a continuous material property function is coupled with the geometry function to generate the graded material cylinder, as the mix of blue and yellow cylinder shown. The specific property function illustrated in Fig. 6.6 varies linearly with the axial distance from the center of capped cylinder. Union operation is non-trivial, usually causing crash in commercial CAD systems. But we can achieve it by applying the methods presented earlier for truss structures like the T-bear in Chapter 4. Then, a calibration step is experimentally measured from light intensity to elastic modulus, and the calibration was employed in computing the property (light intensity) values for the inhomogeneous capsule model. From the pictures of real print below, it is seen that lower light intensity not only affected the material property, but also produced a geometric artifact corresponding to reduction of the capsule cross section. Finally, we got a printed truss as shown below, and it is noticeable that lower gray scale value contributes to incomplete polymerization, thus the smaller radius than the desired one. Thus, our next task below is to address this artifact.

Fig. 6.7 [1] shows additional examples of inhomogeneous octet truss unit cells. The spatially varying property is a processing parameter, the gray scale value to be projected for the pixels in each slice as the truss cell is printed (which in turn controls the degree of polymerization and the elastic modulus). For purposes of illustration, the gray scales are converted to color values with low intensity (70 lx) show in dark blue and high intensity (200 lx) shown in light yellow. For the model shown in Fig. 6.7 (a), each capsule is homogeneous, but those in the mid-plane receive higher illumination, while Fig. 6.7 (b) shows a unit cell where each capsule is inhomogeneous with illumination decreasing with distance from the closest vertex. Fig. 6.7 (c) shows a photograph of the result of printing these models, and the unit cell on the right again clearly shows the artifact of reduced thickness in the center of the capsules where the illumination is low.

To deal with this artifact, cylinders were printed at different illuminations, and their

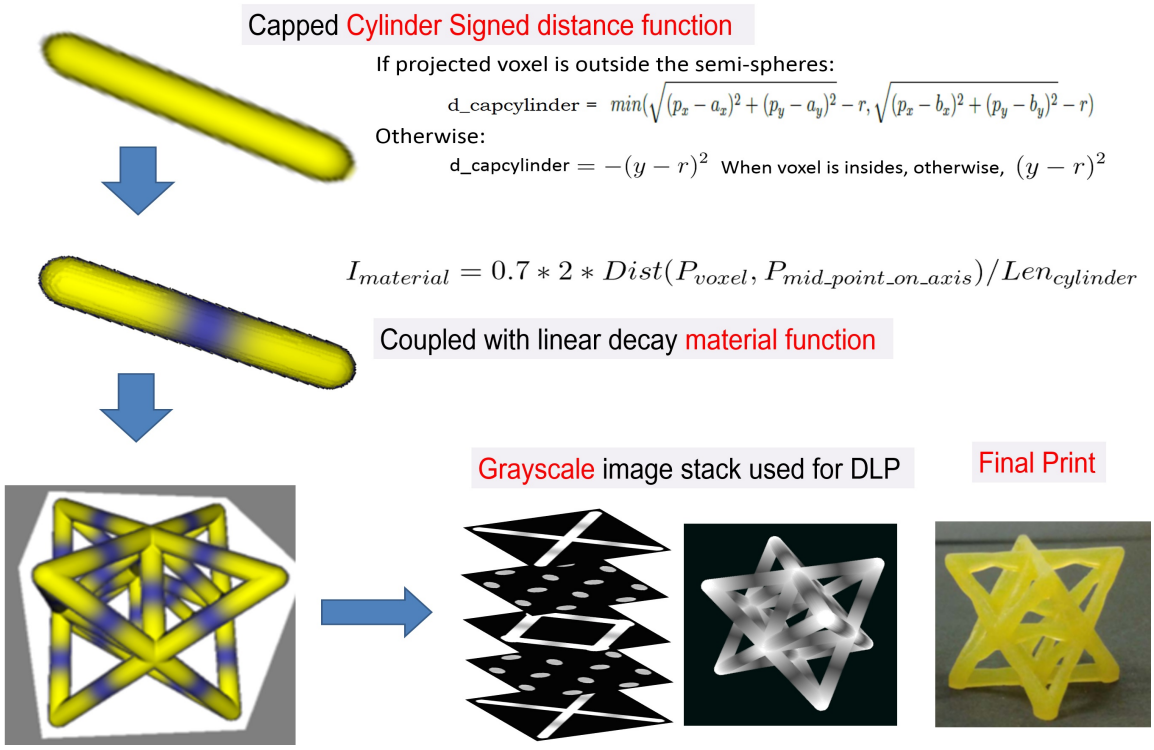

Figure 6.6: The pipeline of generating the whole truss structure consisting of combination of implicit geometry function and material function based graded cylinder bars.

dimensions were measured to determine how to adjust the thickness of the model to maintain the desired output geometry despite variations in light intensity. Since the intensity is designed to vary linearly with distance from the center of the connections between vertices, the connecting capsules were replaced by a union of two cones with spherical end caps. The property function specifying the light intensity remains a linear function of the distance from the mid-plane bisecting the vertices and decreasing from 200 lx at the mid-plane to 70 lx at each vertex. The resulting inhomogeneous discrete SDF-rep model of a connecting element is illustrated in Fig. 6.8.

The octet truss unit cell model is constructed by placing such a capped cone connecting element between each pair of connected vertices and performing a union where the geometry

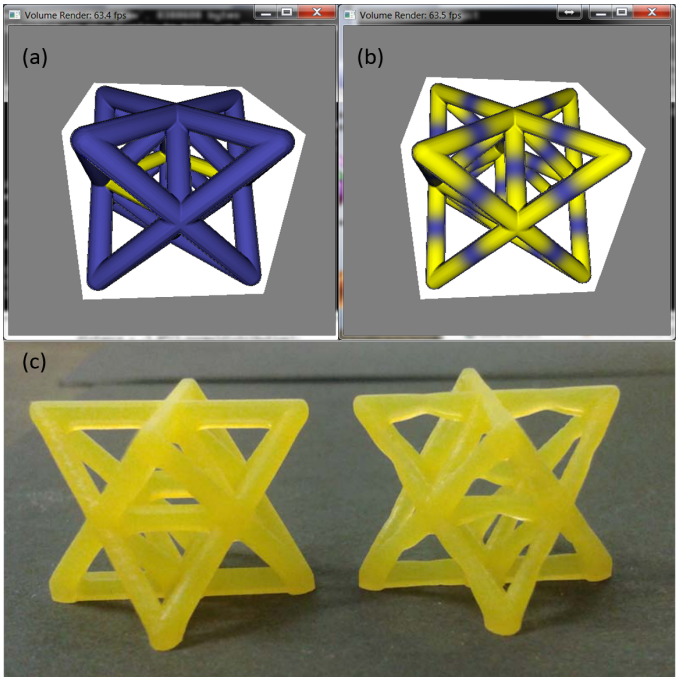

Figure 6.7: (a) homogeneous truss structure with 70 lx (blue) intensity, (b) Heterogeneous truss structure with gradation from intensity of 200 lx (yellow) to 70 lx (blue), while keep the radius the same, (c) real printed two truss, left one corresponding to (a) and right one corresponding to (b). [1]

grid values at each grid point correspond to the minimum of the input values and the property values correspond to the average of the input property values. The intensity-compensated model of the octet truss unit cell is illustrated in Fig. 6.9, and close inspection reveals subtle decreases in thickness at the middle of the connecting elements.

Fig. 6.10 (b) shows the compensated design above the physical output shown on the right side of Fig. Fig. 6.10 (c). The model with intensity-compensated geometry succeed in producing a physical part with connector thickness error of less that 0.5

The newly achieved modeling and fabrication capabilities to design and produce octet truss unit cells with spatially controlled property gradation enables experimental of the mechanical behavior of such structures. Fig. 6.11 [1] show the results of initial testing of 5 versions of the octet truss unit cell. Two of the octet truss cells were homogeneous with light

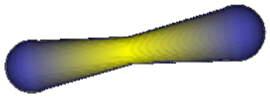

Figure 6.8: Modified signed distance field of two-cone cylinder with various material property.[1]

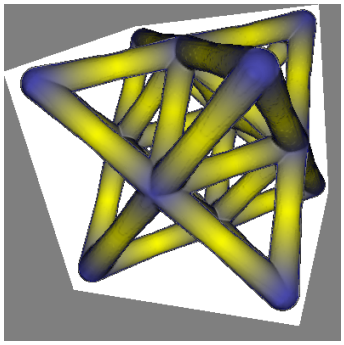

Figure 6.9: Truss structure consists of two-cone cylinders with graded material property.[1]

intensities of 200 lx and 80 lx respectively. The other three truss were inhomogenous (with intensity-compensated geometry) corresponding to: (a) increased intensity in the middle of each connector, (b) increased intensity in the middle of each connector, and (c) low intensity throughout the truss cell except for connectors in the horizontal mid-plane. Fig. 6.11 (b) shows the stress-strain curves measure during Instron compression testing. The first thing to note is that the stress-strain curves differ significantly, so varying the light intensity during the build did produce significant differences in mechanical properties. When looking at the individual curves, it is not surprising that the truss unit cell with uniform high intensity exposure (200 lx indicated by the black curve) supported the highest level of engineering stress. The unit cell with uniform low intensity exposure (80 lx indicated by the red curve)

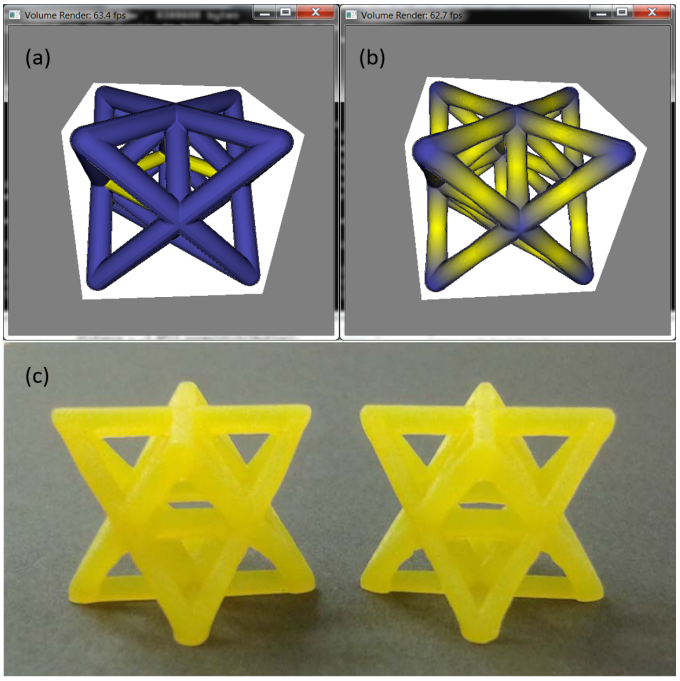

Figure 6.10: (a) homogeneous truss structure with 70 lx (blue) intensity, (b) Heterogeneous truss structure with gradation from intensity of 100 lx (yellow) to 70 lx (blue), and make the radius at both ends of the cylinder larger for radius compensation, (c) real printed two truss, left one corresponds to (a) and right one corresponds to (b). [1]

supported a significantly lower engineering stress, but failed at approximately the same level of engineering strain. Of the graded truss unit cells, case (b) (low intensity in the middle of the connections) showed behavior very similar to the cell with uniform low intensity exposure. Cases (a) and (c), however showed behavior that was significantly different from that of the homogeneous truss cells, and in both cases selective reduction of exposure enabled a trade off between the stress the truss cell can support and the strain it can withstand before failure.

This example serves as proof of concept for inhomogeneous discrete SDF-rep models as the basis for additive manufacturing of parts with spatially controlled, continuously graded properties that significantly affect the large scale mechanical properties of the manufactured part.

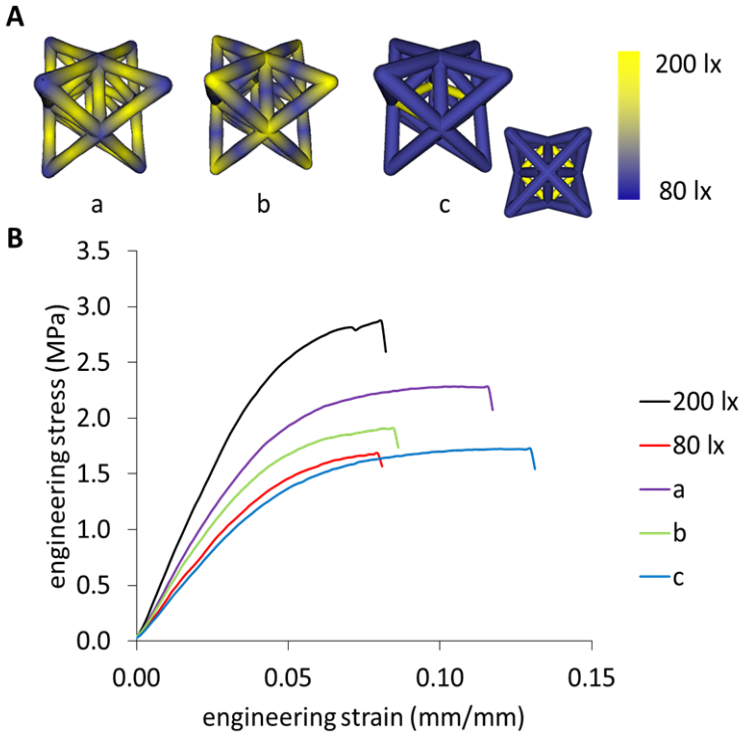

Figure 6.11: (a) Models of the graded and heterogeneous truss structure designs, (b) representative engineering stress/strain curves for all trusses. [1]

Chapter 7

CONCLUSIONS

The significance of data sets involving image stacks or voxel sets is growing due largely to increasing use of 3D imaging systems and 3D printing systems such as powder bed binding and vat photopolymerization. This dissertation describes a research effort to realize a solid modeling system that supports a voxel-based workflow from scan to edit to print. We refer to the basis of the modeling approach as discrete signed distance function representations or discrete SDF which comprise a discrete version of the specific case of implicit or function-based modeling that employs signed distance functions. The underlying data sets are uniform grids of signed distance values that, with an appropriate interpolation method, implicitly define the geometry of solid objects.

Implicit models and, in particular, signed distance function representations have been proposed previously, and discretized versions have also been considered. However, the work described here represents the first implementation of a discrete SDF-rep modeler that supports full interactivity. Moreover, the modeler achieves ground-breaking tools for interactive skeletal editing and modeling of inhomogeneous solids with graded materials, properties, or parameters.

Voxel data sets can be sizeable (a 512 x 512 x 512 grid of 32-bit numbers occupies 0.5 Gigabyte of memory), and there are computational challenges to overcome when trying to interact with large data sets in real time. The discrete SDF-rep modeler addresses these challenges by accelerating all computations using the CUDA system for GPU-based parallelism. High-end computation-oriented GPUs are already available that provide teraflops of compute power and sufficient memory to store dozens of voxel models. The parallel computing power is deployed by the modeler in two significant ways. Two-dimensional computational

grids are launched to support interactive visualization of discrete SDF-rep models. Each parallel computing thread computes the shading for a pixel, and interoperability between CUDA and the OpenGL programming interface for interactive graphics applications enables both real-time display and user interactivity via keyboard, mouse, and menus. For modeling operations, three-dimensional computational grids are launched. Each thread updates the value at a point on the 3D geometric grid, and the local nature of the data interactions (operations typically depend on values at a grid point and possible its neighboring grid points) allows for very effective parallelization. In some cases, special CUDA memory functionalities (shared memories, texture objects, and surface objects) are used to enhance efficiency.

The CUDA-accelerated discrete SDF-rep modeler was found to enable real-time interactions with multiple discrete SDF-rep models and real-time implementations of traditional modeling operations such as rigid body and Boolean operations. As described in Chapter 2, both digitized and digital objects interact readily in the discrete SDF-rep modeler. Demonstrations were provided of importing objects digitized both by volumetric scan (e.g., segmented CT data) and range scan (e.g., point clouds). Digital objects created in a CAD system were imported by exporting STL files and computing grids of signed distance values, and digital SDF were imported by direct function evaluation on a geometric grid.

The discrete SDF-rep modeler implements novel capabilities for creating swept solids by implementing sequences of near-identity rigid body transformations coupled with Boolean operations. Capabilities were realized for real-time sculpting and computation of swept volumes associated with motions of an anatomical structure imported from segmented CT scan data.

The discrete SDF-rep modeler provides interactive computation and visualization of a discrete version of the geometric skeleton. It also includes the first suite of tools for interactive skeleton-based editing and computation of full signed distance grids from skeletal data.

Perhaps the most significant advance achieved by the discrete SDF-rep modeler involves modeling inhomogeneous objects with graded composition or properties. Inhomogeneous discrete SDF-rep models were achieved by appending an auxiliary grid of values specifying

the graded property, and an extended class of Boolean operators were created to combine property values as well as geometry. The novel inhomogeneous modeling capabilities were employed to create models of the unit cell of an octet truss with different patterns of graded properties. The truss structures were fabricated by photo-curing resin in a vat polymerization system using a DLP projector. The mechanism for producing graded properties was to locally control the degree of polymerization through variations in light intensity. The discrete SDF-rep modeler was used to produce the sequences of grayscale images to be used for curing the layers of the truss structures. The truss structures were fabricated to enable mechanical testing, and measurable effect on the loading response of the truss cells were observed. In summary, by exploiting the parallel computing capabilities or modern GPUs the discrete SDF-rep modeler succeeds in supporting an interactive, unified, voxel-based workflow for 3D digital object capture, computer-aided design, and additive manufacturing of objects including heterogeneous objects with graded properties.

BIBLIOGRAPHY

[1] Johanna Schwartz, Greg Peterson, and Di Zhang. Production of materials with spatially-controlled crosslink density via vat photopolymerization. Manuscript submitted for publication, 2016.

[2] Cuda wikipedia website:. `https://en.wikipedia.org/wiki/CUDA`.

[3] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[4] Mark T Ensz. Implicit solid modelling through manifold modification. 1997.

[5] Ben Weiss. convert stl to signed distance field. personal communication.

[6] Luc Soler, Hervé Delingette, Grégoire Malandain, Johan Montagnat, Nicholas Ayache, Christophe Koehl, Olivier Dourthe, Benoit Malassagne, Michelle Smith, Didier Mutter, et al. Fully automatic anatomical, pathological, and functional segmentation from ct scans for hepatic surgery. *Computer Aided Surgery*, 6(3):131–142, 2001.

[7] Yangqiu Hu, William R Ledoux, Michael Fassbind, Eric S Rohr, Bruce J Sangeorzan, and David Haynor. Multi-rigid image segmentation and registration for the analysis of joint motion from three-dimensional magnetic resonance imaging. *Journal of biomechanical engineering*, 133(10):101005, 2011.

[8] Luomin Gao, David G Heath, Brian S Kuszyk, and Elliot K Fishman. Automatic liver segmentation technique for three-dimensional visualization of ct data. *Radiology*, 201(2):359–364, 1996.

[9] Matthew S Brown, Michael F Mcnitt-Gray, Nicholas J Mankovich, Jonathan G Goldin, John Hiller, Laurence S Wilson, and DR Aberie. Method for segmenting chest ct image data using an anatomical model: preliminary results. *IEEE transactions on medical imaging*, 16(6):828–839, 1997.

[10] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005.

[11] A. Capozzoli, C. Curcio, A. Liseno, and S. Savarese. A comparison of fast marching, fast sweeping and fast iterative methods for the solution of the eikonal equation. In *Telecommunications Forum (TELFOR), 2013 21st*, pages 685–688, Nov 2013.

[12] F. Dang, N. Emad, and A. Fender. A fine-grained parallel model for the fast iterative method in solving eikonal equations. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, pages 152–157, Oct 2013.

[13] Miles Detrixhe, Frdric Gibou, and Chohong Min. A parallel fast sweeping method for the eikonal equation. *Journal of Computational Physics*, 237(0):46 – 55, 2013.

[14] H.K. Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 74(250):603–627, 2005.

[15] Hormuz Mostofi, Julia Schnabel, and Vicente Grau. Fast level set segmentation of biomedical images using graphics processing units. *Final Year Project, Keble College*, 2009.

[16] Xi Yang, Xinbo Gao, Jie Li, and Bing Han. A shape-initialized and intensity-adaptive level set method for auroral oval segmentation. *Information Sciences*, 277(0):794 – 807, 2014.

[17] Ozge Oztimur Karadag and Fatos T. Yarman Vural. Image segmentation by fusion of low level and domain specific information via markov random fields. *Pattern Recognition Letters*, 46(0):75 – 82, 2014.

[18] Ross T Whitaker. A level-set approach to 3d reconstruction from range data. *International journal of computer vision*, 29(3):203–231, 1998.

[19] Ron Kimmel and James A Sethian. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision*, 14(3):237 – 244, 2001.

[20] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12 – 49, 1988.

[21] Aly A Farag and M Sabry Hassouna. Theoretical foundations of tracking monotonically advancing fronts using fast marching level set method. Technical report, Technical Report Computer Vision and Image Processing Laboratory, ECE Dept., University of Louisville, 2005.

[22] H.K. Zhao. Parallel implementations of the fast sweeping method. *Journal of Computational Mathematics*, 25(4):421–429, 2007.

[23] Shahnawaz Ahmed, Stanley Bak, Joyce McLaughlin, and Daniel Renzi. A third order accurate fast marching method for the eikonal equation in two dimensions. *SIAM J. Sci. Comput.*, 33(5):2402–2420, September 2011.

[24] M.S. Hassouna and A.A. Farag. Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(9):1563–1574, Sept 2007.

[25] Jakob Andreas Bærentzen. On the implementation of fast marching methods for 3d lattices. Technical report, 2001.

[26] Hongkai Zhao. Parallel implementations of the fast sweeping method. *Journal of Computational Mathematics*, pages 421–429, 2007.

[27] Remondino Fabio et al. From point cloud to surface: the modeling and visualization problem. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(5):W10, 2003.

[28] meshlab official website:. `http://meshlab.sourceforge.net/`.

[29] Atul Kumar, PK Jain, and PM Pathak. Curve reconstruction of digitized surface using k-means algorithm. *Procedia Engineering*, 69:544–549, 2014.

[30] Oliver Schall and Marie Samozino. Surface from scattered points. In *a Brief Survey of Recent Developments. 1st International Workshop on Semantic Virtual Environments, Page (s)*, pages 138–147, 2005.

[31] Jason Sanders and Edward Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[32] Duane Storti and Mete Yurtoglu. *CUDA for Engineers: An Introduction to High-Performance Parallel Computing*. Addison-Wesley Professional, 2015.

[33] Glui official website:. `http://glui.sourceforge.net/`.

[34] Glut official website:. `https://www.opengl.org/resources/libraries/glut/`.

[35] Glui manual website:. `https://www.lri.fr/~mbl/ENS/IG2/docs/glui2.0.pdf`.

[36] Won-Ki Jeong. *Interactive Three-dimensional Image Analysis and Visualization Using Graphics Hardware*. PhD thesis, Salt Lake City, UT, USA, 2008. AAI3338131.

[37] Thomas A. Pitkin III. *GPU Ray Tracing with CUDA*. PhD thesis, Eastern Washington University, 2013.

[38] Charlie C.L. Wang and Yong Chen. Thickening freeform surfaces for solid fabrication. *Rapid Prototyping Journal*, 19(6):395–406, 2013.

[39] Charlie C. L. Wang and Dinesh Manocha. Gpu-based offset surface computation using point samples. *Comput. Aided Des.*, 45(2):321–330, February 2013.

[40] Shengjun Liu and Charlie C.L. Wang. Fast intersection-free offset surface generation from freeform models with triangular meshes. *Automation Science and Engineering, IEEE Transactions on*, 8(2):347–360, April 2011.

[41] James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, July 1982.

[42] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[43] Vadim Shapiro. Real functions for representation of rigid solids. *Computer Aided Geometric Design*, 11(2):153 – 175, 1994.

[44] Tetgen official website:. `http://wias-berlin.de/software/tetgen/`.

[45] Shin Yoshizawa, Alexander G. Belyaev, and Hans-Peter Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, SM '03, pages 247–253, New York, NY, USA, 2003. ACM.

[46] Han-Bing Yan, Shimin Hu, Ralph R. Martin, and Yong-Liang Yang. Shape deformation using a skeleton to drive simplex transformations. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):693–706, 2008.

[47] Justin Wahlborg, Mark A Ganter, Daniel T Schwartz, and Duane Storti. H-ism: An implementation of heterogeneous implicit solid modeling. In *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 973–980. American Society of Mechanical Engineers, 2002.

[48] Manu S Mannoor, Ziwen Jiang, Teena James, Yong Lin Kong, Karen A Malatesta, Winston O Soboyejo, Naveen Verma, David H Gracias, and Michael C McAlpine. 3d printed bionic ears. *Nano letters*, 13(6):2634–2639, 2013.

[49] S. Khalil, J. Nam, and W. Sun. Multi-nozzle deposition for construction of 3d biopolymer tissue scaffolds. *Rapid Prototyping Journal*, 11(1):9–17, 2005.

[50] Li-Hsin Han, Shalu Suri, and Christine E. Schmidt. Fabrication of three-dimensional scaffolds for heterogeneous tissue engineering. *Biomed Microdevices*, 12:721–725, 2010.

[51] Timothy Burg, Cheryl A. P. Cass, Richard Groff, Matthew Pepper, and Karen J. L. Burg. Building off-the-shelf tissue-engineered composites. *Phil. Trans. R. Soc. A*, 368:1839–1862, 2010.

[52] Shoji Maruo, Koji Ikuta, and Toshihide Ninagawa. Multi-polymer microstereolithography for hybrid opto-MEMS. In *The 14th IEEE International Conference on Micro Electro Mechanical Systems*, pages 151–154, 2001.

[53] Mathieu Sanchez. Continuous signed distance field representation. 8 2011.

[54] George Vosselman, Sander Dijkman, et al. 3d building model reconstruction from point clouds and ground plans. *International archives of photogrammetry remote sensing and spatial information sciences*, 34(3/W4):37–44, 2001.

[55] Seng Poh Lim and Habibollah Haron. Surface reconstruction techniques: a review. *Artificial Intelligence Review*, 42(1):59–78, 2014.

[56] autodesk monolith official website:. `http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=13606725`.

[57] autodesk monolith official website:. `https://www.thingiverse.com/`.

[58] myvgl official website:. `http://www.volumegraphics.com/en/products/myvgl/`.

[59] Jan J Koenderink. *Solid shape*, volume 2. Cambridge Univ Press, 1990.

[60] Wikipedia. Superellipsoid — wikipedia, the free encyclopedia, 2014. [Online; accessed 2-December-2015].

[61] V Imbeni, JJ Kruzic, GW Marshall, SJ Marshall, and RO Ritchie. The dentin–enamel junction and the fracture of human teeth. *Nature materials*, 4(3):229–232, 2005.

[62] U Schulz, M Peters, Fr-W Bach, and G Tegeder. Graded coatings for thermal, wear and corrosion barriers. *Materials Science and Engineering: A*, 362(1):61–80, 2003.

[63] Manfred Hofmann. 3d printing gets a boost and opportunities with polymer materials. *ACS Macro Letters*, 3(4):382–386, 2014.

[64] Bethany C Gross, Jayda L Erkal, Sarah Y Lockwood, Chengpeng Chen, and Dana M Spence. Evaluation of 3d printing and its potential impact on biotechnology and the chemical sciences. *Analytical chemistry*, 86(7):3240–3253, 2014.

[65] Nicola Jones. Science in three dimensions: the print revolution. *Nature*, 487(7405):22–23, 2012.

[66] Yong Lin Kong, Ian A Tamargo, Hyoungsoo Kim, Blake N Johnson, Maneesh K Gupta, Tae-Wook Koh, Huai-An Chin, Daniel A Steingart, Barry P Rand, and Michael C McAlpine. 3d printed quantum dot light-emitting diodes. *Nano letters*, 14(12):7017–7023, 2014.

[67] Shaban A Khaled, Jonathan C Burley, Morgan R Alexander, and Clive J Roberts. Desktop 3d printing of controlled release pharmaceutical bilayer tablets. *International journal of pharmaceutics*, 461(1):105–111, 2014.

[68] Yizhou Yu. Surface reconstruction from unorganized points using self-organizing neural networks. In *IEEE Visualization*, volume 99, pages 61–64, 1999.

[69] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[70] PZ Wen, XJ Wu, Y Zhu, and XW Peng. Ls-rbf network based 3d surface reconstruction method. In *2009 Chinese Control and Decision Conference*, pages 5785–5789. IEEE, 2009.

[71] Kun Zhou, Minmin Gong, Xin Huang, and Baining Guo. Data-parallel octrees for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):669–681, 2011.

[72] Tong-guang Ni and Zheng-hua Ma. A fast surface reconstruction algorithm for 3d unorganized points. In *2010 2nd international conference on computer engineering and technology*, 2010.

[73] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421. ACM, 1998.

[74] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[75] Rony Goldenthal and Michel Bercovier. Design of curves and surfaces using multi-objective optimization, 2004.

[76] Yang Liu, Huaiping Yang, and Wenping Wang. Reconstructing b-spline curves from point clouds–a tangential flow approach using least squares minimization. In *International Conference on Shape Modeling and Applications 2005 (SMI'05)*, pages 4–12. IEEE, 2005.

[77] Shengyong Chen, Jianhua Zhang, Houxiang Zhang, Qiu Guan, Yahui Du, Chunyan Yao, and Jianwei Zhang. Myocardial motion analysis for determination of tei-index of human heart. *Sensors*, 10(12):11428–11439, 2010.

[78] Amin Saeedfar and Kasra Barkeshli. Shape reconstruction of three-dimensional conducting curved plates using physical optics, nurbs modeling, and genetic algorithm. *IEEE transactions on antennas and propagation*, 54(9):2497–2507, 2006.

[79] Yao-Chen Tsai, Chung-Yi Huang, Kuan-Yuan Lin, Jiing-Yih Lai, and Wen-Der Ueng. Development of automatic surface reconstruction technique in reverse engineering. *The International Journal of Advanced Manufacturing Technology*, 42(1-2):152–167, 2009.

[80] Farid Boudjemaï, Philippe Biela Enberg, and J-G Postaire. Surface modeling by using self organizing maps of kohonen. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 3, pages 2418–2423. IEEE, 2003.

[81] Fang-Chung Yang, Chung-Hsien Kuo, Jein-Jong Wing, and Ching-Kun Yang. Reconstructing the 3d solder paste surface model using image processing and artificial neural network. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 3, pages 3051–3056. IEEE, 2004.

[82] Joseph T Muth, Daniel M Vogt, Ryan L Truby, Yiğit Mengüç, David B Kolesky, Robert J Wood, and Jennifer A Lewis. Embedded 3d printing of strain sensors within highly stretchable elastomers. *Advanced Materials*, 26(36):6307–6312, 2014.

[83] Ke Sun, Teng-Sing Wei, Bok Yeop Ahn, Jung Yoon Seo, Shen J Dillon, and Jennifer A Lewis. 3d printing of interdigitated li-ion microbattery architectures. *Advanced Materials*, 25(33):4539–4543, 2013.

[84] Duane Storti. Using lattice data to compute surface integral properties of digitized objects. *Proceedings of IDMMEVirtual Concept, Bordeaux, France*, 2010.

[85] Gregory I Peterson, Michael B Larsen, Mark A Ganter, Duane W Storti, and Andrew J Boydston. 3d-printed mechanochromic materials. *ACS applied materials & interfaces*, 7(1):577–583, 2014.

[86] Yong Huang, Ming C Leu, Jyoti Mazumder, and Alkan Donmez. Additive manufacturing: Current state, future potential, gaps and needs, and recommendations. *Journal of Manufacturing Science and Engineering*, 137(1):014001, 2015.

[87] W Pompe, H Worch, M Epple, W Friess, M Gelinsky, P Greil, U Hempel, D Scharnweber, and K Schulte. Functionally graded materials for biomedical applications. *Materials Science and Engineering: A*, 362(1):40–60, 2003.

[88] E Müller, Č Drašar, J Schilz, and WA Kaysser. Functionally graded materials for sensor and energy applications. *Materials Science and Engineering: A*, 362(1):17–39, 2003.

[89] B Kieback, A Neubrand, and H Riedel. Processing techniques for functionally graded materials. *Materials Science and Engineering: A*, 362(1):81–106, 2003.

[90] Ali Miserez, Todd Schneberk, Chengjun Sun, Frank W Zok, and J Herbert Waite. The transition from stiff to compliant materials in squid beaks. *Science*, 319(5871):1816–1819, 2008.

[91] Wikipedia. Additive manufacturing file format — wikipedia, the free encyclopedia, 2015. [Online; accessed 30-November-2015].

[92] Peter H Ernst. Boolean set operations with solid models.

[93] Vibeke Skytt. Challenges in surface-surface intersections. In *Computational methods for algebraic spline surfaces*, pages 11–26. Springer, 2005.

[94] MY Zhou, JT Xi, and JQ Yan. Modeling and processing of functionally graded materials for rapid prototyping. *Journal of Materials Processing Technology*, 146(3):396–402, 2004.

[95] H Liu, T Maekawa, NM Patrikalakis, EM Sachs, and W Cho. Methods for feature-based design of heterogeneous solids. *Computer-Aided Design*, 36(12):1141–1159, 2004.

[96] Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. Openfab: A programmable pipeline for multi-material fabrication. *ACM Transactions on Graphics (TOG)*, 32(4):136, 2013.

[97] Barbara Cutler, Julie Dorsey, Leonard McMillan, Matthias Müller, and Robert Jagnow. A procedural approach to authoring solid models. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 302–311. ACM, 2002.

[98] Todd Robert Jackson. *Analysis of functionally graded material object representation methods*. PhD thesis, Citeseer, 2000.

[99] S. Krishnan, D. Manocha, M. Gopi, T. Culver, and J. Keyser. Boole: A boundary evaluation system for boolean combinations of sculptured solids, 2000.

[100] Carl Lorenz Diener. Procedural modeling with signed distance functions. Master's thesis, Karlsruhe Institute of Technology, Germany, 10 2012.

[101] Aurelien Barbier and Eric Galin. Fast distance computation between a point and cylinders, cones, line-swept spheres and cone-spheres. *Journal of Graphics Tools*, 9(2):11–19, 2004.

[102] T.C. Reiner. Interactive modeling with distance fields. 2010.

[103] P. Dimitrov, C. Phillips, and K. Siddiqi. Robust and efficient skeletal graphs. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 417–423 vol.1, 2000.

[104] Mark T. Ensz, Duane W. Storti, and Mark A. Ganter. Implicit methods for geometry creation. *International Journal of Computational Geometry & Applications*, 08(05n06):509–536, 1998.

[105] Duane Storti, Mark A. Ganter, William R. Ledoux, Randal P. Ching, Yangqiu Patrick Hu, and David Haynor. Wavelet SDF-Reps: Solid Modeling With Volumetric Scans. *Journal of Computing and Information Science in Engineering*, 9, 2009.

[106] M.A.M.M. van Dortmont, H.M.M. van de Wetering, and A.C. Telea. Skeletonization and distance transforms of 3d volumes using graphics hardware. In Attila Kuba, LszlG. Nyl, and Klmn Palgyi, editors, *Discrete Geometry for Computer Imagery*, volume 4245 of *Lecture Notes in Computer Science*, pages 617–629. Springer Berlin Heidelberg, 2006.

[107] Shin Yoshizawa, Alexander Belyaev, and Hans-Peter Seidel. Skeleton-based variational mesh deformations. *Computer Graphics Forum*, 26(3):255–264, 2007.

[108] Duane W. Storti, George M. Turkiyyah, Mark A. Ganter, Chek T. Lim, and Derek M. Stal. Skeleton-based modeling operations on solids. In *Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications*, SMA '97, pages 141–154, New York, NY, USA, 1997. ACM.

[109] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Trans. Graph.*, 27(3):44:1–44:10, August 2008.

[110] Robert Blanding, Cole Brooking, Mark Ganter, and Duane Storti. A skeletal-based solid editor. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, SMA '99, pages 141–150, New York, NY, USA, 1999. ACM.

[111] Tamal K. Dey and Jian Sun. Defining and Computing Curve-skeletons with Medial Geodesic Function. In Alla Sheffer and Konrad Polthier, editors, *Symposium on Geometry Processing*. The Eurographics Association, 2006.

[112] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[113] Mike Roberts, Jeff Packer, Mario Costa Sousa, and Joseph Ross Mitchell. A work-efficient gpu algorithm for level set segmentation. In *Proceedings of the Conference on High Performance Graphics*, HPG '10, pages 123–132, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[114] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[115] Maryam Rastgarpour, Jamshid Shanbehzadeh, and Hamid Soltanian-Zadeh. A hybrid method based on fuzzy clustering and local region-based level set for segmentation of inhomogeneous medical images. *Journal of Medical Systems*, 38(8), 2014.

[116] S. Balla-Arabe, Xinbo Gao, Bin Wang, Fan Yang, and V. Brost. Multi-kernel implicit curve evolution for selected texture region segmentation in vhr satellite images. *Geoscience and Remote Sensing, IEEE Transactions on*, 52(8):5183–5192, Aug 2014.

[117] Duane Storti, Chris Finley, and Mark Ganter. Interval extensions of signed distance functions: isdf-reps and reliable membership classification. *Journal of Computing and Information Science in Engineering*, 10(2):021012–021012–8, 2010.

[118] Howard L. Resnikoff and Raymond O. Wells, Jr. *Wavelet Analysis: The Scalable Structure of Information*. Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[119] Manu S. Mannoor, Ziwen Jiang, Teena James, Yong Lin Kong, Karen A. Malatesta, Winston O. Soboyejo, Naveen Verma, David H. Gracias, and Michael C. McAlpine. 3d printed bionic ears. *Nano Letters*, 13(6):2634–2639, 2013.

[120] Alec Jacobson and Olga Sorkine. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph.*, 30(6):165:1–165:8, December 2011.

[121] Sukmoon Chang. Extracting skeletons from distance maps. *IJCSNS International Journal of Computer Science and Network Security*, 7(7), 2007.

[122] Martin Rumpf and Alexandru Telea. *A Continuous Skeletonization Method Based on Level Sets*. University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science, 2002. Relation: http://www.rug.nl/informatica/organisatie/overorganisatie/iwi Rights: University of Groningen. Research Institute for Mathematics and Computing Science (IWI).

[123] O. Rouiller. Real time rendering of skeletal implicit surfaces. Master's thesis, Technical University of Denmark, DTU Informatics, E-mail: reception@imm.dtu.dk, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2011. Supervised by Associate Professor Jakob Andreas Bærentzen, jab@imm.dtu.dk.

[124] David B Kirk and W Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Newnes, 2012.

[125] Pitchaya Sitthi-Amorn, Javier E. Ramos, Yuwang Wangy, Joyce Kwan, Justin Lan, Wenshou Wang, and Wojciech Matusik. Multifab: A machine vision assisted platform for multi-material 3d printing. *ACM Trans. Graph.*, 34(4):129:1–129:11, July 2015.

[126] Jonathan D. Hiller and Hod Lipson. Design automation for multi-material printing.

[127] E.L. Doubrovski, E.Y. Tsai, D. Dikovsky, J.M.P. Geraedts, H. Herr, and N. Oxman. Voxel-based fabrication through material property mapping: A design method for bitmap printing. *Computer-Aided Design*, 60:3 – 13, 2015. Material Ecology.

[128] Vinod Kumar, D Burns, Debasish Dutta, and C Hoffmann. A framework for object modeling. *Computer-Aided Design*, 31(9):541–556, 1999.

[129] Chek T. Lim. *Reconstruction of Solids from Surface Data Points Through Implicit Functions*. PhD thesis, Seattle, WA, USA, 1997. UMI Order No. GAX97-36323.

[130] Alexander Pasko, Oleg Fryazinov, Turlif Vilbrandt, Pierre-Alain Fayolle, and Valery Adzhiev. Procedural function-based modelling of volumetric microstructures. *Graphical Models*, 73(5):165–181, 2011.

[131] Mario Botsch and Leif Kobbelt. A robust procedure to eliminate degenerate faces from triangle meshes. In *VMV*, pages 283–290, 2001.

[132] Oleg Fryazinov, Alexander Pasko, and Valery Adzhiev. Bsp-fields: An exact representation of polygonal objects by differentiable scalar fields based on binary space partitioning. *Computer-Aided Design*, 43(3):265–277, 2011.

[133] Bradley A Payne and Arthur W Toga. Distance field manipulation of surface models. *IEEE Computer graphics and applications*, (1):65–71, 1992.

[134] Galina Pasko, Alexander Pasko, and Tosiyasu Kunii. Space–time blending. *Computer Animation and Virtual Worlds*, 15(2):109–121, 2004.

[135] Mark W Jones. 3d distance from a point to a triangle. *Department of Computer Science, University of Wales Swansea Technical Report CSR-5*, 1995.

[136] Nikhil Gagvani and Deborah Silver. Parameter-controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999.

[137] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[138] Susan Fisher and Ming C Lin. *Deformed distance fields for simulation of non-penetrating flexible bodies*. Springer, 2001.

[139] Arnulph Fuhrmann, Gerrit Sobotka, and Clemens Groß. Distance fields for rapid collision detection in physically based modeling. In *Proceedings of GraphiCon 2003*, pages 58–65, 2003.

[140] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 871–878. ACM, 2003.

[141] E. Napadensky. Compositions and methods for use in three dimensional model printing, May 2003. US Patent 6,569,373.

[142] Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph.*, 29(4):63:1–63:10, July 2010.

[143] Yue Dong, Jiaping Wang, Fabio Pellacini, Xin Tong, and Baining Guo. Fabricating spatially-varying subsurface scattering. *ACM Trans. Graph.*, 29(4):62:1–62:10, 2010.

[144] Leon S. Dimas, Graham H. Bratzel, Ido Eylon, and Markus J. Buehler. Tough composites inspired by mineralized natural materials: Computation, 3D printing, and testing. *Advanced Functional Materials*, 23(36):4629–4638, 2013.

[145] L. Wang, J. Lau, E. L. Thomas, and M. C. Boyce. Co-continuous composite materials for stiffness, strength, and energy dissipation. *Advanced Materials*, 23(13):1524–1529, 2011.

[146] Neri Oxman. Variable property rapid prototyping. *Journal of Virtual and Physical Prototyping (VPP)*, 6(1):3–31, 2011.

[147] Jonathan D. Hiller and Hod Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2012.

[148] Thomas A. Campbell and Olga S. Ivanova. 3D printing of multifunctional nanocomposites. *Nano Today*, 8(2):119 – 120, 2013.

[149] Stratasys. Objet500 Connex Multi-Material 3D Printer http://www.stratasys.com/3D-Printers/design-series/precision/objet-connex500.

[150] Jae-Won Choi, Ho-Chan Kim, and Ryan Wicker. Multi-material stereolithography. *Journal of Materials Processing Technology*, 211(3):318–328, 2011.

[151] Asim Inamdar, Marco Magana, Frank Medina, Yinko Grajeda, and Ryan Wicker. Development of an automated multiple material stereolithography machine. In *Proceedings of 17th Annual Solid Freeform Fabrication Symposium, Austin, TX*, pages 624–635, 2006.

[152] Pranav Kumar, James K. Santosa, Elizabeth Beck, and Suman Das. Direct-write deposition of fine powders through miniature hopper-nozzles for multi-material solid freeform fabrication. *Rapid Prototyping Journal*, 10(1):14–23, 2004.

[153] Yanxiang Lan, Yue Dong, Fabio Pellacini, and Xin Tong. Bi-scale appearance fabrication. *ACM Trans. Graph.*, 32(4):145:1–145:12, 2013.

[154] Wojciech Matusik, Boris Ajdin, Jinwei Gu, Jason Lawrence, Hendrik P. A. Lensch, Fabio Pellacini, and Szymon Rusinkiewicz. Printing spatially-varying reflectance. *ACM Trans. Graph.*, 28(5):128:1–128:9, 2009.

[155] Tim Weyrich, Pieter Peers, Wojciech Matusik, and Szymon Rusinkiewicz. Fabricating microgeometry for custom surface reflectance. *ACM Trans. Graph.*, 28(3):32:1–32:6, 2009.

[156] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. Inverse bi-scale material design. *ACM Trans. Graph.*, 32:163:1–163:10, 2013.

[157] B. Derby. Inkjet Printing of Functional and Structural Materials: Fluid Property Requirements, Feature Stability, and Resolution. *ANNUAL REVIEW OF MATERIALS RESEARCH*, 40, 2010.

[158] Wonjoon Cho, Emanuel M Sachs, Nicholas M Patrikalakis, and Donald E Troxel. A dithering algorithm for local composition control with three-dimensional printing. *Computer-aided design*, 35(9):851–867, 2003.

[159] Fujifilm Dimatix. Dimatix materials printer dmp-2831.

[160] Justin Lan. Design and fabrication of a modular multi-material 3d printer. Master's thesis, Massachusetts Institute of Technology.

[161] Jocye Kwan. Design of electronics for a high-resolution, multi-material, and modular 3d printer. Master's thesis, Massachusetts Institute of Technology.

[162] Javier Ramos. Multifab: A multi-material 3d printing platform. Master's thesis, Massachusetts Institute of Technology.

Appendix A

RELATED BACKGROUND

A.1 Overview of Traditional Modelers and Motivation for Adopting Distance Fields

Currently, there are many modeling systems available in the market, but the market share is dominated by boundary representation or b-rep modelers that describe a solid object in terms of a collection of surface patches that connect to form the boundary of the solid volume. The b-rep category includes polyhedral models whose boundaries can be decomposed into a triangle mesh. Triangle mesh models are convenient because they can be stored and rendered very efficiently; consumer graphics hardware has been created and optimized for the purpose of rendering triangle meshes. While triangle meshes can be rendered with shading or texturing to appear smooth, the triangles in the mesh are flat and cannot really represent surfaces with smooth, contoured surfaces. In the context of 3D printing or additive manufacturing, tactile inspection of parts reveals the polygonization artifacts regardless of surface shading or texture. One approach to achieving models with smooth surfaces is to use b-reps with more complicated parametric surface patch descriptions. Options include bicubic, bezier, and non-uniform rational b-splines (NURBS) which currently hold a significant market share. These parametric methods store model boundaries in a memory efficient fashion and generally offer good flexibility and precision. However, higher order parametric surface representation might be inefficient in terms of computational complexity when rendering or doing boolean operations, and it is challenging to import digitized objects (obtained by range scan or volumetric scan) into a b-rep modeler. Moreover, since even smooth b-rep models generally get polygonized before export to an AM system, obtaining physical realizations of smooth b-rep models is still problematic. The other issue in the AM context is that b-rep

models are really designed to describe the surfaces that bound a solid; such models are not really designed to describe internal variations of properties and materials that AM systems are capable of producing. Thus, the motivation to consider alternative representations such as discrete SDF-reps consists of 3 main components: (1) to create an environment where digital CAD models can readily interact with digitized or scanned objects; (2) to provide support for models surfaces that are smooth, at least to the resolution of an AM system; and (3) to support representation of smooth gradations of internal materials and properties.

In light of the motivating factors listed above, implicit or function-based (f-rep) models are an attractive alternative to b-rep models. An f-rep model consists of a mathematical expression that can be evaluated at a given point in the 3D modeling domain to classify the point as interior or exterior. We employ the sign convention that negative function values identify interior points. The boundary of an f-rep solid is the zero level set of the function and, since it is well known how to write mathematical functions with at least piece wise smooth level sets, f-reps can readily model objects with smooth, contoured surfaces. Since f-reps involve a function defined throughout the modeling domain, not just on the surface of an object, they also offer excellent prospects for representing properties throughout the interior of a solid. Thus f-reps offer immediate promise for achieving 2 of the 3 motivating factors, but it is not so obvious that they are suited to the goal of supporting import of digitized objects.

The specific version of f-reps that we pursue is a discretized version where the function is specified by a set of values on a regular geometric grid along with an interpolant for computing values at points not belonging to the grid. We also focus on the specific case fo signed distance functions where the magnitude of the function value gives the distance between the evaluation point and the nearest point on the boundary of the object (while the sign of the function value classifies the point). As discussed in Ch. 2, this discrete signed distance function representation (discrete SDF-rep) successfully supports import of objects based on continuous f-rep models, b-rep models, surface scan point clouds, and segmented volumetric imaging. It has been reported that rendering and construction of

signed distance field based representation is possible for complicated geomteryies and relevant classes of solid shapes[104, 123]. Some researches report using signed distance functions for real time collision detection[139], either rigid solids[140] or soft bodies[138]. Surface offsetting operation is easily done by manipulating distance fields[134, 135]. Signed distance fields approach is also suitable and straightforward to perform boolean operation efficiently[137, 105] without CSG (constructive solid geometry) operations on b-rep models, and apply complicated 3D texturing[104]. adopt a discrete approach and evaluate the signed distance function on a 3D grid, so the data set resembles an image stack or a voxel set. Unlike traditional voxel models where entire voxels are classified and the surface is only known to lie within a set of unclassified voxels with axis aligned faces, this voxel data can take on continuous values that are interpolated to implicitly define boundaries for which the surface normal vector and curvature properties (determined by the rate of change of the normal vector) are well-defined and readily computable. As discussed in Ch. 6, a similar approach to handling property specifications by interpolation a grid of values supports modeling solids with locally-controlled, continuously-graded materials, properties, or processing parameters.

To import 3D models from other sources (b-reps, surface scans, and volumetric scans), we need to classify points and compute surface distance values on a grid. Computing signed distance values for a polygonal mesh can be computationally expensive if both the number of polygons and the number of grid points is large. An alternative approach to obtain the signed distance field of a 3D object involves volumetric digitization using a scanning method such as CT. The CT scanner gives the intensity of radiation, but does not directly provide point membership classification. Therefore, a further segmentation step is required to identify the voxels inside the models. A method for obtaining high resolution model of 3D objects based on volumetric information has been reported by using wavelets to interpolate SDF models[105]. Furthermore, to make the SDF modeler promising, several studies using signed distance functions for procedural modeling and simple geometries achieved in real time have been recently presented[100, 101, 102].

There are actually several other voxel based softwares available in the market, including 2

worthy of special attention: (1) myVGL [58], is a volume graphics viewer for volumetric data from CT scanner. While myVGL provides some segmentation capabilities, it is intended as a viewer, and the companion software VGStudio is required to get access to a limited set of basic modeling operations. (2) monolith [56] is a voxel modeler with a very original user interface and that offers a reasonable set of modeling operations including boolean operations and sweeping constructions. However, monolith does not attempt to support models imported from other sources and appears to have resolution limitations associated with memory usage.

A.2 Overview of Approaches about 3D Model Reconstruction from Point Cloud

It is useful to people, who wants to reversely reconstruct the 3D model [54] from surface scanned data, or point cloud. Currently, there are four main ideas [55] in the graphics literature about 3D reconstructing the surface of objects: signed distance estimation, voronoi-based reconstruction, implicit surface fitting and moving least squares surfaces. Challenges faced by all of these methods include reconstructing models without surface normals; reconstructing small or sharp features, and dealing with noisy data and outlier points..

The surface scanned point cloud is one type of input data set for 3D reconstruction process, which basically only stores the coordinates of all the points. However, reconstructing a 3D model from the unstructured input data set is difficult without the mesh topology among all the points, and it is hard to guarantee that the reconstructed surface matches the original shape of the objects since there is no information provided regarding the correct connectivity of the sample points [68, 81, 80], expecially when it comes to dealing with the curve net (a network representing the curved surface) construction as well as surface fitting [79]. Therefore, the first step is always figuring out the correct connectivity among all the cloud points. Basically, there are two different types of surface representation: explicit surface and implicit surface. Explicit surface like B-spline surface [77, 76] or NURBS surface (parametric surfaces) [75] describes the location of a surface [78] using formulas, while the triangulated

surfaces is kind of approximation of original surface of objects. However, the triangulated surfaces approache needs to figure out the mesh topology or connectivity among the cloud points, usually by forming nearest neighbor connections[74, 73, 72]. While, for implicit surface, sometimes also called volumetric representation or function based representation, it represents the surface as a special iso-contour of a scalar function [129]. The signed distance function, radial basis function, moving least square [70] are those techniques most used for implicit forms [71].

A.3 3D Skeleton Capturing, And Manipulating

Recent developments in solid modeling have frequently focused on local and global free-form skeleton-based deformation approach due to its capabilities of offering a simpler and more intuitive interface to users. The skeletal data, a lower dimensional geometric abstraction for performing geometric operations on solid models[108], offers a method for solid representation based on skeletal data consisting of skeletal points and their associated maximal sphere radii.

Reported methods for computing geometric skeletons include applying medial geodesic function[111], extracting skeletons from distance maps[121] based on level sets[122], or presampling the mesh model into a volumetric representation [103], using graphics hardware for skeletonization[106]. Another recently reported approach applies Laplacian smoothing to contract the boundary of a volume onto its skeleton [109]. Afterwards, a variety of methods can be applied on the meshed or polygonal models for deformations by editing the skeleton data. Once skeleton computation is available, skeletal modeling operations can be supported. Skeleton-based algorithms have been proposed for: level of detail control, hexahedral mesh generation, shape interpolation and morphing, shape synthesis[108] to achieve stretching, bending, rounding as well as editing maximal sphere radii of the models based on the continuous skeleton[110]. For real time deformation applications, skeleton-based linear blend skinning, also known as skeletal subspace deformation remains standard for character animation[120], shape interpolation as well as skeleton-driven deformation[112, 107, 123]. In terms of mesh models deformation, compared with the method of fixing the location of an ar-

bitrary vertex to locate the deformed mesh relative to the skeleton, a better solution reported is to incorporate a translation term in the error energy function to determine the deformed mesh configuration[46]. A recent report proposed a new scheme for free-form skeleton-driven mesh deformations[45] where the voronoi-based skeletal mesh is extracted from the original mesh, then deformed skeletal meshes contribute to a desired global shape deformation.

A.4 Multi-material 3D printing

Objects with multi-material or graded-material heterogeneously distributed spatially throughout the whole body is ubiquitous in nature. For instance, common graded material includes the teeth, bones, and squid beaks [61, 90], where the material transits from soft to hard spatially. Besides, interesting objects with various material property like reflectance is presented [154, 155]. Mimicking these graded-material property holds great potential for development of medical material or other engineer material for beams or truss structures [62, 87, 88, 145]. However, unfortunately, synthesis of bulk multi-material or graded material using traditional material-removal manufacturing requires tedious extra steps and still heavily limited by geometries [89]. Therefore, an efficient manner for development of multi-material or graded material products holds tremendous future both on design and manufacturing sides. 3D printing, also called additive manufacturing, attracts lots of interests due to the nature of building product from ground layer by layer, versatile for specifying certain material composition for different parts in the objects spatially[146, 148]. Thus, 3D printing is ideal for developing those nature mimicking graded material products[86, 63, 64, 65, 85, 66, 67, 82, 83], various texture[143], or Bi-scale appearance fabrication[153, 156]. MIT researchers presented a work enables to construct multiscale synthetic material[144]. And similar ideas have extends to micro scale 3D printing as well using stereography[150, 151], deposition of fine powders through hopper nozzles[152].

With traditional polygon based CAD system, it lacks the capability of specifying graded material throughout the b-rep models, which needs to decouple the material function from the geometry itself. Here, we present a voxel based SDF-rep CAD system which naturally

matches well with the additive manufacturing in terms of the fact that both of them specify and build for each voxel(in 3D) or pixel(in 2D) during the process. However, the huge amount of voxles of 10^8 poses an enormous computational challenge, and until recently the General Purpose GPU card available for parallel computing makes it doable in our life time.

Most of the 3D printing machines in the market use only one single material at a time, there are still several 3D printers (e.g., Object Connex series [114, 149]) enabling users to print emerging multi-material but only with certain mixed ratio without gradation. In terms of the software, until recently, 3D printing has been using uniform material definition to produce objects based on boundary representation (polygon based) CAD system with unstructured meshes of the surfaces. Although various companies have establised their own formats[114, 149, 159] to accomplish 3D multi-material or graded material printing capability, the common methods of assigning a single material property to each point of b-rep models always contributes to the limitations of designing spatial material property decoupled with geometry (multi-material information needs to be in separated STL files). In addition, some work has reported that sub-division spatially on mesh data structures can achieve local high resolution result[98]. Although, the Additive Manufacturing File Format (AMF) standard [91] decouples the material information from geometry, it is still handling with b-rep models, thus slicing procedure from mesh models is inevitable (which takes the most of the time when exporting the models file to machine instructions).

The advantage of designing multi-material function or graded material function decoupled from models' geometry is that we can get different desired material compositions and thus mechanical properties for different geometries. For example, we can print an "I" beam with two kinds of materials, the stronger one can be applied on bottom and top parts for bearing the bending stress, while the less strong material can be applied to the middle part used to connect two functioning parts (top and bottom parts). Of course, the analysis requires the FEM (finite element method) under certain situation or specific boundary condition [126] for compensation of deformation[142, 147] due to the stress, heat effect and so on. Nevertheless, the manufacturing process needs a suitable CAD system for specifying and

precisely manipulating the material functions so that the fabricated objects can possess desired functionalities. Kumar reported a method to describe material composition throughout the objects spatially by subdividing volumes into sub-volumes, where local material interpolation is based on coordinates[128, 141]. Recently, the procedural material assignment representations is presented to describe material composition for solid models[97]. Even more, a programmable pipeline for describing material composition precisely and conveniently throughout a 3D objects is published[96]. A further exploration [125, 160, 161, 162] works on integrating self-calibrated printheads, 3D scanning, and close feedback loop for printing correction multi-material 3D printing platform with high resolution and low cost.

Due to the difficulties in modeling multi-material or graded material objects in traditional b-rep CAD system, prior work explored the possibility and efficiency of volumetric representations based on voxels [98, 127]. Furthermore, some work reported that a 2D or 3D dithering methods can be used to define anisotropic fluid material using inkjet printing [95, 94, 157, 158]. And one of those important prior works also used a signed distance filed to represent geometry of objects while using composition function for specifying material property throughout the objects spatially. Based on these amazing works, we push the voxel modeler to a higher level, where we can not only define material function on voxel based discrete signed distance field, but also using the graded geometries to build more complex structures like sweeping single graded object to get a swept graded union object. Besides, our voxel modeler is compatible with digital light projection system 3D printing since we directly editing voxel information (or pixel in 2D), therefore, directly outputting images layer by layer without any extra slicing effort.

A.5 Overview of GPU Architecture and Programmable Graphics Hardware

Over the past decade, semiconductor industry designing follows two main directions. The first one is multicore trajectory, for example the recent Intel Core i7, which using multiple cores to execute sequential programs. These multi processors are implemented with the X86 instruction set, designed to optimize the the execution speed of sequential programs. Instead

of focusing on maximizing execution speed on sequential programs, the other approach, called many -thread trajectory adopts the idea of maximizing the throughput. The many-threads approach needs lots of threads, in our case, we have maximum of 2147483647*65535*65535 threads for Tesla K20 GPU cards from the manul. In 2012, the ratio of peak floating point calculation throughput between many thread GPUs and multicore CPUs is about 10[124]. This ratio makes GPU parallel computing promising on the well parallel-able computing intensive applications. Actually, GPU usually serves as co-processor with CPU, called heterogeneous architecture. CPU and GPU are physically separated and communicate with each other through PCI express bus, while the GPU holds its own memory (DRAM) as well as lots of multiprocessors consisting of processor and its on-chip memory (registers, shared memory and caches)[36].

All processors shared in the same multiprocessor execute the same instruction at the same time within the single multiprocessor. Different processors can communicate with each via shared memory, while multiprocessors can communicate with each other via DRAM. Since the bandwidth of the PCI express bus is much smaller than that between the GPU multiprocessors and GPU DRAM, the efficient strategy should copy data from CPU side to GPU DRAM and using the shared memory, register, constant memory or texture memory to optimize the execution to maximize the throughout of parallel applications.

Since some computation requires to access the neighborhood data, we need to pack the data on the GPU memory in a certain way so that we have access to the memory as coalesced as possible. There are several approaches achieving the coalesced data, for instance compacting data from sparse array format or rearranging array of structure (AOS) to structure of array (SOA). Here, instead of using shared memory, which is only readable within the same block, or constant memory, which cannot be allocated dynamically, we employ texture memory on the GPU which enables to sample the voxel information on an arbitrary location efficiently. However, CUDA texture objects are read only, which is not desirable for our updating scheme. Thus, another similar architecture with 3D space neighborhood coalesced memory called surface memory, which is not only readable but also writable, is adopted in

our algorithm. The initial boundary adopted here could be either the enclosed surface when it comes to the Boolean operations, or the skeleton data when it comes to "refleshing" from the skeleton data. In detail, the GPU fast sweeping method first stores all the 3D voxel signed distance field into a surface memory with tagging the boundary value points as fixed while tagging other voxels as changeable, meanwhile assigning all the initial inactive grid points with an infinite vale. During each update, before it is convergent to a threshold, the GPU fast sweeping method algorithm keeps updating both unknown grid points(if it is an infinite value) as well as those points already explored but with changeable tags. During each iteration, the updated 3D grid data from one surface memory needs to be saved to another surface memory for next round update. An example below shows the performance comparison when using CPU and GPU, along with different memory types in GPU card.

A.5.1 Comparison of CPU and GPU when Computing the Gradient

Instead of adopting CPU serial version, here, we implement GPU parallel version to calculate the gradient of the given signed distance field. Essentially, for each grid point, the gradient is estimated by the central difference scheme along all three principle axis. Instead of adopting CPU scheme with a few cores optimized for sequential serial processing, our GPU scheme has a massively parallel architecture consisting of thousands cores designed for handling multiple tasks simultaneously, which contributes to hiding latency by using more threads per multiprocessor. Therefore, in users' perspective, GPU parallel version is usually faster than the CPU serial version especially when the data set is huge enough and the application is parallel-able well. Below, is an experiment conducted to compute gradient for the grids of 64∗64∗64, 128∗128∗128, and 256∗256∗256 respectively on CPU (Intel core i5-3330 3.00GHZ) as well as on GPU (Tesla K20c). The Table A.1 records the timing of CPU serial version and GPU parallel version with texture memory to computing gradient on those three different grids.

From the Table A.1, it is clearly seen that the GPU parallel version is more efficient and faster to get all the gradient computing done for all the grid points. Besides, it is also

Table A.1: Time taken for computing gradient of distance field by CPU and GPU

Grid dimension	CPU time	GPU time	CPU/GPU
64∗64∗64	11.0 ms	1.0 ms	11
128∗128∗128	91.0 ms	2.7 ms	33.7
256∗256∗256	683.0 ms	18.0 ms	37.9

interesting to observe the fact that with the grid dimension increasing, more advantage we can take by using GPU parallel version with the help of texture memroy compared with CPU serial version, concluded from the column of CPU time over GPU time in Table A.1.

A.5.2 Optimization of memory fetching

Furthermore, there are several different memory types available on GPU parallel version, for example, share memory, global memory, texture memory and constant memory. Only when using the appropriate memory type under the certain circumstance, the application can be accelerated to the maximum degree. Since all these three scenarios are parallelized, the efficiency difference actually results from the cache hitting. Therefore, the sections below are to explore different memory types in terms of cache hitting rate and try to explain the strategy we adopted in our applications.

Global memory Global memory is also called device memory, which resides in device DRAM, for transferring the data between the host and device as well as for the data input to and output from kernels. In terms of memory coalesce, once we are fetching certain data stored as an array in global memory, several other data whose physical location in memory close to that specific data are also fetched and stored together in the cache. Therefore, next time whenever we need fetch the data whose physical location in memory locality close to the pre-fetched data, the system will directly get data from the cache instead of reading from global memory again, which accelerates the application and is called cache hit. Another thing

needs to be noticed is that the cache hit rate is depressed in our 3D space for computing gradient. Because we need to get the central difference in three orthogonal axis, and among those six adjacent neighbor grid points in 3D space, only three of them are actually linearly offset locality to each other, indicating only two of them might hit the cache. Therefore, we desire a different kind of memory type which could cache the coalesced data in 2D and 3D space locality.

Shared memory Now, it is time to explore the shared memory, which is on chip memory, closer to the processor compared with global memory. Thus, the shared memory is much faster than global memory, roughly 100x lower than un-cached global memory latency. Therefore, loading data directly from shared memory definitely will be faster than fetching from global memory. The only problem is that we need to store all the data from global memory into the shared memory at the very beginning as initialization. This step could be extra time-wasting if the data is only used once in each block later, or time-saving if each data in the shared memory is repeatedly used in each block. Therefore, unless fully taking advantage of the low latency of shared memory when computing gradient, it will not help a lot. Another subtle issue for the shared memory when computing the gradients is that it needs to deal with the special situation when computing the gradient on the block boundaries, where the neighbor stencils have to be fetched from another block or the global memory. What is worse, there has to be the if-else statement to deal with the boundary conditions which also contributes to the multi-branching problems, depressing the parallel scheme. Besides, the limited share memory on chip makes it inappropriate in our modeling system since we need to store large data set, 128*128*128 gird, which can not be handled by share memory.

Texture memory Due to the slower data fetching of global memory and limited share memory, we further explore the texture memory, which has advantages over previous two memory types. Texture memory, a read only memory, is for general purpose computing the

data need to be fetched in 2D or 3D stencil pattern. Texture memory is also cached on chip like shared memory. Therefore, similar to constant memory, it will provide higher effective bandwidth by reducing memory requests to off-chip DRAM. More specifically, texture memory is designed for graphics applications where memory access patterns exhibit 2D or even 3D spatial locality, which can never be achieved by global memory caching(linear locality in 1D). In this work, we stick to employ the texture memory cache to fetch all the spatial locality data and pre-store them in texture objects. However, when we use upwind differencing scheme to recover the signed distance field, the texture memory can not rewrite the update distance value on grid points. Fortunately, there is a writable surface memory similar to texture memory, in terms of fetching the 3D spatial locality data, used for recovering signed distance field.