The Dissertation Committee for Lalit Chandra Sekhar Karlapalem certifies that this is the approved version of the following dissertation:

A Volumetric Sculpting Based Approach for Modeling Multi-Scale Domains

Committee:

Richard H. Crawford, Supervisor

Matthew I. Campbell

Ofodike A. Ezekoye

Carolyn C. Seepersad

Kamy Sepehrnoori

A Volumetric Sculpting Based Approach for Modeling Multi-Scale Domains

by

Lalit Chandra Sekhar Karlapalem, B.S.;M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2006

Dedicated to my Parents and my Wife.

Acknowledgments

I wish to thank the multitudes of people without whose help I would not have reached this far.

I extend my heart felt gratitude to my parents without whose efforts, help and constant support I would never have made it this far. It is your undying belief in me, in the face of severe adversity that has given me a ray of hope and led me to complete several tasks without giving up. Words are not enough to express my feelings for your nurturing and for all the great memories that you have provided me with.

I want to thank my wife Madhuri, for her endless love and understanding through the last several years. Her cool composure and sound advice have really eased a lot of pressure from me.

I am eternally grateful to several faculty and staff at The University of Texas at Austin, who took time out from their busy schedules to meet with me and understand my problems and guide me through them. It is my good fortune that I had close and long associations with more than a few such faculty/staff during my stay in UT.

I do not know where to begin when expressing my thanks to my advisor Dr. Richard Crawford. In the 6 years that I have known him, I have gained a wealth of knowledge from him. His cheerful personality never failed to lift my

spirits in the darkest of situations and make me see the bright side of things. He has constantly challenged me to think outside the box and come up with innovative ideas and pursue them.

Dr. Ofodike Ezekoye has been more than a committee member and graduate advisor. His timely helpful suggestions have saved the day for me on more than one occasion and I am really indebted to him.

One gloomy thought is of my rather very short alliance with Dr. Kamy Sepehrnoori. He has constantly gone out of his way to help me with both personal and academic matters. Irrespective of his constant tight schedule and my problems, I never needed an appointment and I have never left his room disappointed. He has always made me feel welcome in his research group and I would dearly miss working with him.

The constant encouragement and helpful suggestions given by Dr. Campbell have really made me look forward to my meetings with him. Thank you for that. I would also like to thank Dr. Seepersad for helping me to narrow the scope of my research.

I also want to say Nihau to Jianguang 'Jgsun' Sun and Dr. Yongjie 'Jessica' Zhang and thank them for the friendly and fun filled work atmosphere in CCV. I also want to extend my thanks to Dr. Vinay Siddavanahalli for his constant help during my entire stay in Austin.

I want to appreciate and thank Dr. Francisco Marcondes for taking time out from his busy work schedule to answer my never ending questions

and help me stay on track.

Finally, I would like to thank all the friends that I have made in Austin without whose help, I would not have so many fond memories and good experiences of my stay here.

A Volumetric Sculpting Based Approach for Modeling Multi-Scale Domains

Publication No. _____

Lalit Chandra Sekhar Karlapalem, Ph.D.
The University of Texas at Austin, 2006

Supervisor: Richard H. Crawford

This dissertation addresses the problem of creating computer simulations for multi-scale domains. These domains span a wide range of both spatial and temporal scales. Creating realistic simulations based on computer models for such domains requires very large memory overhead and computing resources.

A framework for modeling such multi-scale domains is described here. The proposed approach creates an initial geometric scaffold (grid) for the domain. This scaffolding can be further (locally) refined to closely match the geometric features of the domain. Functions can be defined on this grid to model different physical parameters of interest. These piecewise linear functions are interpolated within a cell to provide uniform global C^0 functions over the complete domain. Of particular interest are the geometry and physics

functions that are formulated over this scaffold, which enable geometric and computational modeling of the domain.

Given a set of processes of interest on the domain, the next step focuses on creation of a computational model of the domain by defining another set of functions over the same scaffold. Finally, the physics model is numerically solved over the domain geometry in an attempt to simulate the physical process.

To demonstrate that the approach works, a case study of the classical problem of heat conduction through porus media is presented. For this, high resolution geometric models for a powder bed in the Selective Laser Sintering (SLS) process are developed. This comprehensive multi-scale model is then validated with a simulation of the percolation process to study the effect of heat conduction through the media. The key phenomenon of interest is the manner in which the heat flows through the domain containing a given configuration of the composite.

The main focus of this research is to provide a robust and comprehensive framework for creating realistic computational and geometric multi-scale models. These models are then interleaved to yield a comprehensive numerical simulation of the process.

Table of Contents

Acknowledgments v

Abstract viii

List of Tables xiii

List of Figures xiv

Chapter 1. Introduction 1
- 1.1 Multi-Scale Domain Modeling Pipeline . . . 3
- 1.2 Problem Statement . . . 6
  - 1.2.1 Research Objectives . . . 6
  - 1.2.2 Research Scope . . . 7
- 1.3 Overview of Dissertation . . . 8

Chapter 2. Background and Previous Work 10
- 2.1 Preliminaries . . . 10
- 2.2 Geometric Modeling Tools . . . 12
  - 2.2.1 Volumetric Representations . . . 12
  - 2.2.2 Signed Distance Function Computations . . . 14
  - 2.2.3 Volumetric Sculpting . . . 15
    - 2.2.3.1 Sculpting Operations . . . 19
  - 2.2.4 Evolving Surfaces . . . 20
  - 2.2.5 Improving Mesh Quality . . . 21
  - 2.2.6 Visualization Techniques . . . 23
- 2.3 Computational Physics Modeling . . . 24
  - 2.3.1 Types of Domain Models . . . 25
    - 2.3.1.1 Continuum Models . . . 25
    - 2.3.1.2 Discrete Formulations . . . 25

2.3.2 Multi-Scale Computational Modeling 26
2.3.3 Heat Conduction Continuum Model 27
2.4 Case Study Domain: Selective Laser Sintering 29
2.4.1 Heat Transfer Mechanisms for SLS 29
2.4.2 Modeling Heat Percolation 31
2.4.2.1 Theoretical Models 31
2.4.2.2 Representative Element Volume Based Models . 32
2.4.2.3 Empirical Models 33
2.5 Summary 33

Chapter 3. Geometric Modeling **35**
3.1 Creating Volumetric Representations 35
3.2 Signed Distance Field Computations 38
3.3 Scale Bridging 39
3.4 Generating the Intermediate Volumes 42
3.5 Volumetric Sculpting 43
3.6 Volumetric Domain Representations 44
3.7 Surface Roughness Models 46
3.8 Summary 50

Chapter 4. Computational Modeling **51**
4.1 Finite Difference Models 52
4.2 Stability Analysis 54
4.3 Modeling Initial and Boundary Conditions 56
4.4 Steady State Simulations 58
4.4.1 Boundary Conditions at the Coarse-Fine Grid Interface 59
4.5 Transient State Simulations 59
4.5.1 Boundary Conditions at the Coarse-Fine Grid Interface 61
4.6 Heat Conduction Models 61
4.6.1 Steady State Simulations 62
4.6.2 Transient State Simulations 62
4.7 Summary 64

Chapter 5. Application Domain: Selective Laser Sintering **65**
5.1 Introduction 65
5.2 Unit Cell Models 67
5.3 Modeling Surface Roughness 68
5.4 Implementation Details 69
5.5 Time and Space Complexity Analysis 71
5.5.1 General Time and Space Complexity Requirements . . . 71
5.5.2 Spatial Storage 72
5.5.3 Time Complexity 73
5.5.4 Analysis 74
5.6 Summary 75

Chapter 6. Closure **76**
6.1 Conclusions 76
6.2 Future Work 77
6.2.1 Geometric Modeling 77
6.2.2 Computational Modeling 78
6.2.3 Volumetric Mesh Extraction 78

Appendix **80**

Appendix A. Modeling Boundary Conditions **81**
A.1 Taylor Series 81
A.2 Steady State Heat Conduction Model 84
A.3 Transient State Heat Conduction Model 85

Bibliography **87**

Vita **103**

List of Tables

4.1 Stability Analysis for constant mesh ratio m 55
4.2 Stability Analysis for constant Fourier number r 56
4.3 Simulation values for 3D heat conduction simulation 64

5.1 Results for ETC (W/m-K) for Polycarbonate-Air bed 67

List of Figures

1.1 An example of the different temporal and spatial scales in nature [23]. 2

1.2 The multi-scale modeling pipeline. 4

2.1 The different modes of heat transfer are shown in the powder bed here. [72] 30

3.1 The sketch of the hierarchical data structures used for scale bridging. 41

3.2 Algorithm for the volumetric sculpting routine 45

3.3 Surface roughness models with different values of σ and grid resolution. 48

3.4 Comprehensive geometric domain representation showing 3 levels of hierarchy. 49

4.1 Finite difference scheme at the boundary. 53

4.2 Neumann boundary conditions on a curved boundary. The sign change edges are shown in blue and the actual points of intersection in red. The surface normals at these points are computed and shown in green. At point H, the normal is flipped and intersected with the grid at point E. 57

4.3 Results from the heat conduction simulation. 63

5.1 The multi-scale domain model for the body centered cubic dataset. Two contact regions are shown with multi-scale representations. 68

A.1 Finite difference scheme at the boundary. Consider the 3D neighborhood of point P. The boundary points A, B, C, D, E and F are at distances $\alpha\Delta x$, $\beta\Delta x$, $\gamma\Delta y$, $\delta\Delta y$, $\lambda\Delta z$, $\mu\Delta z$, respectively from P. 82

Chapter 1

Introduction

Many natural phenomena occur over varying ranges of length and time scales. Some atomic phenomena inside living bodies occur at the time scale of a few femtoseconds (10^{-15}), while other galactic processes occur at the time scale of a few million years. Refer to Figure 1.1. Such domains are called *multi-scale domains* and their models are called *multi-scale models*. Similarly, in the spatial domain, the atomic scale is of the order of a few Angstroms, while most of the daily objects in use are at the metric scale. While good and accurate models are available for most of these processes, the problems arising from creating realistic and comprehensive computer simulations are not often completely solved by using single scale models. One big limitation is the accuracy of the solutions offered by single scale models. As one desires better accuracy, multi-scale models begin to offer a wider range of options with improved accuracy. Furthermore single scale models completely neglect the effect of changes occurring at the other scales. Thus, by coupling models at different scales, it is possible to merge their relative advantages and present a more unified, comprehensive and realistic model. Some typical examples are listed below:

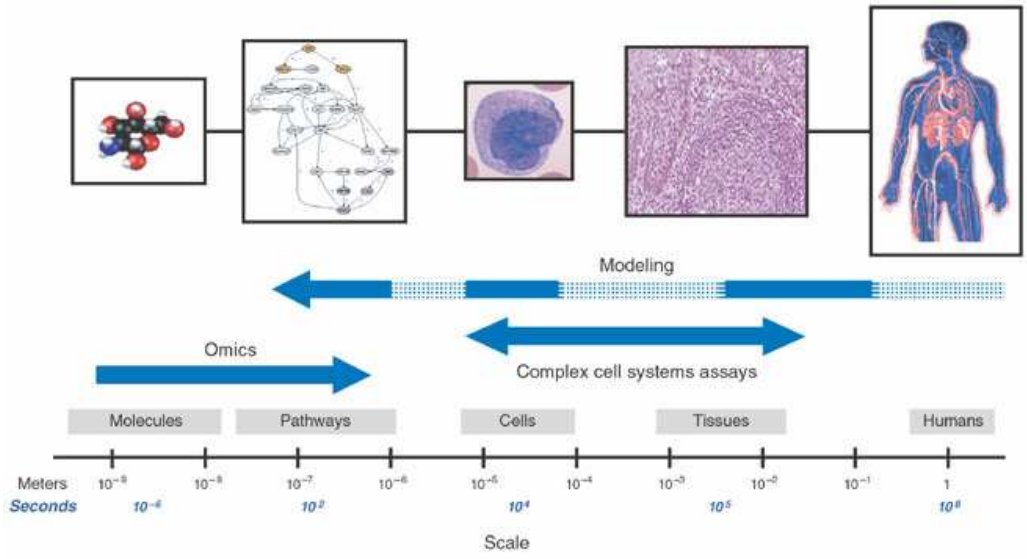

Figure 1.1: An example of the different temporal and spatial scales in nature [23].

- Nanotechnology: With giant strides in technology in the recent years, fabrication of products at the nanoscales is being actively pursued and might soon become a reality. The basic laws of classical physics such as Newton's laws are not applicable to nano particles. Techniques from quantum mechanics are needed to deal with these phenomena [34].

- Computational Biology: Consider the classical problem of protein folding. The vibrations of the covalent bonds are of the order of femtoseconds, while the protein folding process itself is of the order of seconds.

- Computational Physics: Seismology is an important branch that deals with the analysis and prediction of earthquakes. At the macro-scale, the focus is on the physics and deformation and the general magnitude of

the quakes. At the micro-scale, research focuses on fracture and rupture analysis [34].

With recent advances in computational power, multi-scale problems can be solved. These often require the generation of a good mesh as a pre-processing step, over which the rest of the computations are formulated and solved. Traditional approaches include multi-grid methods, domain decompositions [99], adaptive mesh refinement techniques [109] and multi-resolution methods using wavelets [34]. All of these have been developed to tackle the problem of large scale data.

1.1 Multi-Scale Domain Modeling Pipeline

Figure 1.2 shows the conceptual design of a multi-scale modeling pipeline. Starting from the input specifications, first the volumetric representations are generated as the common data format for the models. Then, using volumetric sculpting operations, the final volumetric mesh for the given domain is created. As indicated, there are two major stages in the pipeline. The different types of input data are first converted into volumetric representations. Then, volumetric sculpting operations are performed to create a geometric domain model. This is the physically based geometric modeling phase. Then, the computational model is created for this domain and solved on the geometric model. Thus, a simulation is performed and the model is used to make physical predictions. Volumetric and boundary (surface) meshes can also be output from the representations, as desired.

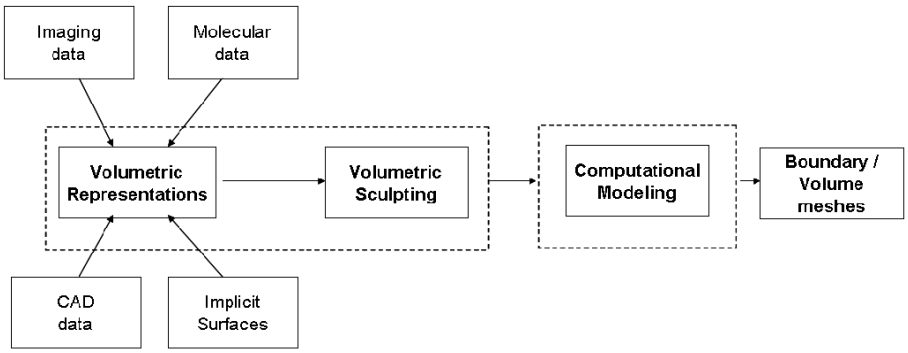

Figure 1.2: The multi-scale modeling pipeline.

The input data could be specified in any of the following formats:

- Imaging data: This data is often obtained from Magnetic Resonance Imaging (MRI), Computerized Tomography (CT), Electron Microscopy (EM) or CryoEM, and is in the form of 2D slices.

- Molecular data: A molecule is a set of atoms. Each atom is specified by its center and radius and is approximated as a sphere. In addition to this, the secondary and tertiary structures are specified to completely represent a molecule. All this information is present in the popular PDB file format [19] for each molecule.

- CAD data: Here, the input models can be in the form of Constructive Solid Geometry (CSG), Boundary Representations (B-reps) or Non-Uniform Rational B-Splines (NURBS), to name a few [45, 59]. They can be readily converted into tesselated geometry, typically triangles. Popular examples include StereoLithography (STL) files and Initial Graphics Exchange Specification (IGES) formats.

- Implicit Surfaces: The geometry can be specified by a set of implicit surfaces. In some cases, this is useful, as the complete domain could simply consist of multiple copies of the same object. A typical example could be a box of embedded particles, with each particle modeled with a function (e.g. a sphere).

In a typical multi-scale domain, one set of geometry describes the macro-scale object, while a second set(s) of geometry describes the embedded fine features in the domain. These two geometries are at different length scales. So, the domain geometry needs to be interactively edited and the *features* placed on it. This operation is called *Volumetric Sculpting*.

Once the comprehensive domain geometric representation is created, an appropriate physics model that can be applied is then identified. This is achieved by defining another set of physics functions over the volumetric representations that are used to model the geometrical entities of the domain. Finally, a numerical discretization is performed and computationally solved.

As a by-product of the geometric representation, high quality meshes can also be output if needed. It is worthwhile to mention that these meshes are not directly used in the simulations, but are used only for visualization purposes. Using direct contouring methods, both surface and volumetric meshes can be extracted from the framework. These finite element meshes can then be used for other applications as desired.

1.2 Problem Statement

The focus of this research is to establish a new framework for modeling the geometry of multi-scale domains in a manner that supports simulation of physical phenomena at multiple scales. The primary goal is to construct a comprehensive geometrical representation of the domain, using volumetric sculpting based techniques. Then, a computational model is constructed using this geometrical representation to perform simulations for a given physics domain.

1.2.1 Research Objectives

The principal research objectives are as follows:

Given a surface S and a set of surfaces of arbitrary genus C_i with different length scales, and a function F describing the distribution of C_i on S, the goal is to first construct a geometric representation of the complete domain.

Next, a set of partial differential equations governing the physical process of interest is described as part of the computational model and solved numerically on the geometrical domain representation. Specifically, the research focuses on the following three areas:

- Geometric modeling: First the set of input macro-scale and micro-scale geometry needs to be converted into a common representation. Then,

these models should be used to construct a comprehensive domain representation using volumetric sculpting based techniques.

- Computational modeling: The resulting representation of the domain needs to be validated with the domain functionality. This can be checked by numerically solving a set of (partial) differential equations governing the physical process in the domain, thus simulating the actual intended application for the product.

1.2.2 Research Scope

As mentioned in Section 1.1, the framework is specially suited for first creating geometrical representations for domains that span multiple length scales. The representation scheme is inherently recursive, intuitively hierarchical, spatially adaptive and has great flexibility.

In general, depending upon the particular application domain, usually different sets of physics models are proposed at different length scales (e.g. oil/gas reservoir simulations). This means that one function needs to be defined for each of the physical parameters in the models (one function for each *degree of freedom* or *state variable*). Conversely, it is also possible for domains to have only one physics model to describe the entire range of length scales (e.g. heat conduction).

While the proposed framework is general and can handle a wide range of applications, the current research scope is restricted to one sample application

domain. The various assumptions for the current implementation are listed here.

- Single physics model: only those domains that can be completely described by a single physics model are identified and used. This simplifies the implementation for the numerical solver. This is also the primary reason that the scope is restricted to heat conduction through porous media.

- Steady State Analysis: This is a rather direct consequence of the chosen application domain. Only the steady state simulations are considered in the research. However, Section 4.5 outlines the steps that are needed to adapt the framework for analyzing transient state simulations.

As mentioned above, these assumptions can be relaxed readily to create a general purpose framework.

1.3 Overview of Dissertation

The remainder of the text is arranged as follows. In chapter 2, an overview of the relevant previous work is presented. Specifically, the areas of geometric modeling and computational modeling are covered along with a brief discussion of the chosen application domain.

Chapter 3 describes the details of the geometric modeling phase of the framework. The input data (in any of the forms described in Section

3.1) is first converted into volumetric representations which are then sculpted together to form the multi-scale volumes. This provides a hierarchical multi-scale volumetric grid that is then used as the basis for subsequent simulations.

In chapter 4, the basics of numerical simulations based on computational modeling of domains are covered. Then, a means to customize numerical techniques to the proposed framework is shown, along with the definition of appropriate functions needed for the simulations.

Chapter 5 shows the application of the framework to the domain of selective laser sintering and addresses implementation details.

Future extensions to the current framework are discussed in chapter 6 along with conclusions.

Chapter 2

Background and Previous Work

In this chapter, state of the art approaches in *geometric domain modeling* and *multi-scale computational modeling* are presented. This is followed by a closer look at the selected case study domain of Selective Laser Sintering (SLS).

2.1 Preliminaries

First, a few fundamental concepts and basic terms used in this dissertation are defined.

Domain: For the purpose of this dissertation, a domain is defined as any real product(s) governed by a set of physical principles. Of specific interest are the domains in R^3 space, particularly those which span across several length scales.

Manifold: A topological space $M \subseteq R^m$ is called a *manifold* if at each and every point in the space $P \in M$, there exists an open set $O \in M$ such that: $P \in O$ and the open space O is homeomorphic to R^n and n is fixed for all the points in M. Then the $n-$dimensional manifold is said to be embedded in the $m-$dimensional space. Implicit in the definition is the condition $m \geq n$

[68]. From this definition, it follows that a *2-manifold* (or surface) is said to be embedded in R^3, if every point on it is homeomorphic to a plane.

Grid: A lattice structure consisting of uniformly spaced points along the (local or global) co-ordinate axis directions is called a *grid*. If the spacing between incident points is the same along all directions, then it is an *isotropic grid*, otherwise it is an *anisotropic grid*.

Mesh: A mesh provides a discretization of the domain using a set of nodes and elements with strictly defined connectivity, such that the union of these elements gives a complete and unambiguous representation of the domain. The mesh elements could be linear or higher order entities. For a given domain in R^3, it is possible to have a *surface mesh* which typically contains polygonal elements and a *volumetric mesh* which consists of polyhedral elements. These shapes provide only a linear shape description across each element.

Computational modeling: A mathematical model is formulated to predict the behavior of the domain under the prescribed physical conditions.

Research Assumptions: For the sake of this dissertation, the vast range of potential applications where the framework can be used has been restricted. Some of these limitations are imposed by the choice of representations, while others greatly simplify the modeling (both geometric and computational) efforts.

- Domain: The governing computational model for the domain can contain

at most one physical parameter. Also, a single computational model should be able to predict the performance of the domain. This eliminates the need to synchronize more than one computational model for every iteration of the simulation (See Sections 2.3.2 and 4.5 for details.)

- 2-manifold: Since volumetric representations are extensively used for the shape representations, this restricts the geometric models to only closed surfaces, where the *boundary* clearly separates the interior from the exterior. Such objects are also called *solid models*.

2.2 Geometric Modeling Tools

Implicit surfaces are represented in a compact mathematical form in a volumetric domain [22]. Examples of implicit surfaces include *iso-surfaces*, *level sets*, *Gaussian surfaces* and *blobby models*. In this text, of primary concern are the first two representations. Since both of them require a background in *volumetric representations*, this and related concepts are closely examined first.

2.2.1 Volumetric Representations

The volumetric domain is represented by a 3D array of points with a function defined at each point. In the simplest form, these points can be associated with a binary value indicating whether they are inside or outside the object to be modeled [45, 59]. However, this leads to aliasing, where a straight line would be represented by a jagged sequence of voxels [96]. A

better approximation of the object is achieved by storing a set of scalar values that are sampled from a *characteristic function* $f : R^3 \rightarrow R$. A standard example of this is the *Signed Distance Function* (SDF) from the volumetric grid point to the boundary surface of the solid [40] (Please also see Section 2.2.2).

$$F(x,y,z) = F_{000}(1-x)(1-y)(1-z)+F_{001}(1-x)(1-y)(z)+F_{010}(1-x)(y)(1-z) + F_{100}(x)(1-y)(1-z) + F_{011}(1-x)(y)(z) + F_{101}(x)(1-y)(z) + F_{110}(x)(y)(1-z) + F_{111}(x)(y)(z) \quad (2.1)$$

Thus, a function is defined over this grid such that the "zero" of the function yields the original boundary surface (also called the *iso-surface*) [58]. This is shown in Equation 2.1 [57]. The accuracy of the surface reconstruction depends on the grid resolution and surface polygonization algorithms [5, 40]. Recently, *vector* signed distance fields [50] have also been used, while Frisken et al. [40] have experimented with using *adaptive* distance fields to improve the computational cost and efficiency. *Marching cubes* algorithms, which were first proposed by Lorensen et al. [58], have been used to extract structured meshes from volumetric representations. Primal contouring [53] has been used to extract good quality structured meshes. Dual contouring [49] yields adaptive hierarchical meshes based on an error tolerance. It is also possible to extract tetrahedral meshes from the above representations as shown by Ning et al. [71]. However, the volumetric representations using distance fields are

particularly suited for hexahedral mesh generation and hence research has focused on this problem. Schneiders [86, 87] provides templates for creating conforming hexahedral meshes from adaptive representations, and Zhang et al. [109] extend this scheme to generate high quality meshes. Bajaj et al. [9] have used subdivision based schemes to extract optimal error bounded mesh elements.

2.2.2 Signed Distance Function Computations

Computing the signed distance function is a very crucial first step in generating the volumetric representation of various datasets. Here, the focus is on computing the SDF for a given tessellated closed surface.

The *brute force* approach demands the computation of the actual Euclidean distance for each grid point of the volume representation from the given surface. This algorithm is $O(NM)$ in computational cost, where N is the number of grid points and M is the number of surface polygons. In practice, N is very high, depending upon the resolution of the volumetric grid, and can be as high as $1878 \times 1728 \times 800$ or about *15 gigabytes*, as in the case of the *visible human* dataset [95]. Hence, this is a *very* computationally expensive approach.

Previous work has focussed on mainly two approaches to speed up these computations:

- Voronoi Diagrams: First, the Voronoi diagram is constructed for the

input domain geometry consisting of the points, edges and polygons. Then each grid point is located in a Voronoi 3D cell and the closest distance to the polygons in those cells can be computed easily. Thus, the Voronoi diagram acts as a spatial partitioning technique and then the Euclidean distance is computed locally. The overall time complexity of the algorithm is $O(N^2)$ in the worst case and $O(NlogN)$ in the average case [9, 62].

- Distance Transforms: The actual Euclidean distance for the boundary voxels is first computed. Then, this distance is *propagated* to the rest of the interior voxels using distance transforms. These can be further categorized as *chamfer distance transforms* and *vector distance transforms*. Both work in the same way, except that in the former case, the scalar distance values are propagated, while in the latter, the vector components are propagated [84, 85].

With recent advances in graphics hardware, research has also focused on performing some computations on the Graphics Processing Unit (GPU), instead of the CPU [8, 93]. Programming directly on the graphics card results in accelerated rendering. This has improved the overall performance manyfold.

2.2.3 Volumetric Sculpting

Freeform modeling started in the mid 1980s by Sederberg et al. [89] as a new paradigm for modeling shapes. The discretized volume space is

represented by small continuous lattices (uniform cells). The desired surface patch inside each of these lattices is defined by a *tensor product tri-cubic Bezier volume* as:

$$F(u,v,w) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} P_{ijk}B_i(u)B_j(v)B_k(w) \tag{2.2}$$

with $0 \leq u, v, w \leq 1$, the $B_i(t)$ are the degree 3 *Bernstein Polynomials* given as $B_i^m(t) = \binom{m}{i}t^i(1-t)^{m-i}$, and P_{ijk} are the control points. These control points are defined for each such surface patch and they provide local control of the surface inside the control (sub)volume. The immediate problem with the approach was that the manipulation of these control points soon became cumbersome and in some cases, too complex. Also, some skill is required on the part of the user to manipulate the control points to get the desired shape. Coquillart [29] provided the additional flexibility of configuring the initial lattice structure to better approximate the surface. Instead of having to deal with the massive array of control points, Hsu et al. [46] provided a simple framework where the user performs basic operations on a sample set of points and these are applied to the total surface (or some local region of interest). Implicit functions are used to interactively model complex surfaces with optimal modeling elements [22]. Skeletons of standard primitive shapes are built and interactively modified using weighted blending functions based on control points [27].

In the early 1990s, volumetric sculpting was first introduced by Galyean

et al. [41] as a new freeform modeling technique for interactively modeling topologically complex shapes[1]. While the approach was intuitive to the user, it could not guarantee accuracy, precision or close tolerances. Starting from a voxelized grid containing scalar values at the grid points (similar to the volumetric representations described in Section 2.2.1), they provide the user with a suite of tools that can change the function values at these grid points, thus changing the shape of the resulting iso-surface. Their work was based on previous work for implicit surface modeling by Bloomenthal [22] and Coquillart [29].

Here, the solid object is represented by a 3D raster of voxels. Specific local properties like shape (topology), size (geometry), look and feel (object properties like texture and color) and material properties (like density, elasticity, stiffness etc.) can be stored in each voxel. However, in practice, in most cases, instead of storing all of this information, only a scalar value like the *intensity* [41] (for images) or the *signed distance function* [82] (for objects) is chosen. The rest of the features like the object properties (e.g. color, texture) and material properties (e.g. elasticity) can be defined using *transfer functions* [36, 37]. McDonnell et al. [64] use this concept extensively to perform interactive sculpting in their software *DigitalSculpture*.

There has been considerable interest in volumetric sculpting since it was first introduced in the early 1990s. Wang et al. [103] extended the original

[1]Please refer to [83] and [59] for a good survey of the existing modeling techniques in the late 1980's.

approach to provide more sculpting operations and interactivity to the user. Raviv et al. [82] represented the sculpted object as the zero set of a scalar tri-variate B-spline function. Sculpting was performed by changing the values of the coefficients of the basis functions. For rendering, they polygonized the function. Ferley et al. [36] extended the sculpting paradigm to include freeform based tool shapes and focused on improving the rendering quality using texture maps. Frisken et al. [80] created the volumetric models using an *Adaptively sampled Distance Field (ADF)* which creates optimal meshing elements in their popular software called *Kizamu* for sculpting digital characters. Chandru et al. [26] have extended the concept to Solid Freeform Fabrication (SFF), using the volumetric representations to create solid models of complex shapes. Singh et al. [94] use skeletons as a first approximation to the object and then specify the deformations along the mesh elements therein to produce a real-time volume modeling environment. Rappoport et al. [81] use a set of solid primitives to approximate the object. Then, they recast the local deformations on each of these as a global optimization problem, whose solution yields the desired boundary surface while preserving the enclosed volume of the solid.

The volumetric representation makes it easy to think of an object as blocks of material, from which material can be progressively removed in different forms to yield the desired shape. The tools required by these sculpting operations are also represented by a similar 3D array of voxels. In the most naive case, only binary values indicating the shape of the tool can be used [41]. However, to reduce *aliasing* [96], a volumetric representation similar to

the one described above for the object is also used for the tool [14, 82, 103] but at a higher resolution [36, 103]. The tool can be moved around interactively in the 3D volume of the object. When the tool is engaged, the associated operation is performed on the two overlapping volumes.

2.2.3.1 Sculpting Operations

Broadly, the various sculpting operations can be categorized as follows:

- Subtractive operations: These operations remove the material from the object and include:

  1. *Sawing* : This tool removes regular block shapes from the volume, like a carpenter removing a chunk of material.
  2. *Carving* : Here, the tool shape is arbitrary. The volume is *carved* with this tool to replicate its shape in the object. Carving and Sawing are the stock operations used in sculpting.
  3. *Heat Gun* : The tool *melts* away the material.

- Additive operations: These add material to the object. These operations include:

  1. *Painting* : This tool can change the local function value at a voxel. This is used for finer control.
  2. *Squirt* : This is like squeezing toothpaste out of a tube.

3. *Pasting* : This can add material to the volume. The amount is proportional to the shape of the tool.

- Smoothing operations: These operations are used to remove the aliasing effects caused by discretizing smooth surfaces [96]. Smoothing tools do not have any associated function values. With a smoothing tool, each voxel within the tool's range is replaced by a weighted average of its current value and those of its six neighbors (1^{st} *ring neighbors*). Other anti-aliasing functions like Gaussian filters [36] and Catmull-Rom splines [96] have also been used.

Cutler et al. [30] have described a scripting language based on physically based modeling for sculpting characters on surfaces.

2.2.4 Evolving Surfaces

Often, a first approximation to the desired surface can be modeled. The desired surface can then be generated by iteratively refining this initial surface, such that in the limit the desired (smooth) surface can be obtained. This could be done by simply using a subdivision scheme such as Loop's subdivision scheme [56] or Multi-Linear-Centroid-Smoothing (MLCS) [11]. Subdivision meshes are a popular scheme for defining a mesh. For an excellent survey of different subdivision schemes, please refer to the tutorial by Zorin et al. [88].

Research has also addressed starting with a much closer approximation (to the desired surface) and then defining some set of Partial Differential Equa-

tions (PDEs) on this domain. Local and global control is possible by changing the boundary conditions and the *controlling functions*. Classical examples include using Poisson equations for smoothing [107] and non-linear equations for surface modeling [104].

Two or more meshes can also be merged to form a new mesh. Yu et al. [107] define a Poisson equation to evolve the mesh globally and also modify it locally by changing the gradient fields. DeCarlo et al. [32] provide a hierarchical deformable framework where meshes can be blended gradually to yield the desired mesh. They use a transformation to and from the object space into the parametric space, where they do their blending.

2.2.5 Improving Mesh Quality

Mesh quality is an important aspect of mesh generation. Often meshes can be rendered ineffective because of the huge number of computations and memory overhead associated with them. Quality mesh generation has indeed been an important topic over the last few decades and a lot of research has focused on this [20, 92]. Bern et al. [20] and Berzins [21] provide good surveys of the different aspects of mesh quality. Here, some of the criteria for defining and improving mesh quality are presented.

- Error Tolerance: The error between the mesh boundary (surface) and the *true* surface must be kept within a tight bounds [28]. Usually, this can be accomplished by using higher order elements or by further subdividing the mesh to form smaller elements that can then be repositioned to

reduce the error. Standard subdivision schemes like Loop's scheme [56], Kobbelt's Butterfly scheme [50], or Bajaj et al.'s Multi-Linear Centroid Smoothing Scheme (MLCS) [11] can be used for refining the mesh.

- Aspect Ratios: The aspect ratio is defined as the ratio of the length of the maximum edge of an element to the length of the minimum altitude [92]. Large (and small) values of aspect ratios can lead to degeneracies in the solver, resulting in incorrect results and longer running times [20].

- Condition Numbers: If $\overline{x} \in \Re^3$ is the position vector of a grid vertex and $\overline{x_i} \in \Re^3$ for $i = 1, ..., m$ are its neighboring vertices, then an *edge vector* is defined as $\overline{e_i} = \overline{x_i} - \overline{x}$ with $i = 1, ..., m$ and the *Jacobian* $J = [\overline{e_1}, ..., \overline{e_m}]$ [109] (for triangular/quad meshes, m=2 and for tetrahedral/hexahedral meshes m=3). The determinant of the Jacobian matrix is called the Jacobian or scaled Jacobian, if the edge vectors are normalized. An element is *inverted* if its Jacobian ≤ 0. The *condition number* of the Jacobian is defined as $\kappa(J) = |J||J^{-1}|$, where $|J^{-1}| = \frac{|\widehat{J}|}{det(J)}$ and $|\widehat{J}|$ is the adjoint of the Jacobian. The goal is to remove the inverted elements and improve the worst condition number [9].

- Mesh Optimality: It can be shown that the running time of a finite element analysis is $O(n^a)$, where n is the number of mesh elements and a is at least one, depending upon the numerical solver used. Mitchell [66] gives a sample theoretical proof for the number of tetrahedral elements in a mesh. He has a scheme that is guaranteed to converge with bounded

aspect ratios for its elements.

- Robustness: Finite element meshes used to model critical components need to be robust [21]. The algorithms for mesh generation often produce degeneracies due to the floating point errors which are inherent in the discretizations and in the computations. Topological inconsistencies arise from these degeneracies and need to be dealt with [20, 21].

- Feature Adaptivity: The embedded features in the mesh are very important for both computation and visualization purposes. In some applications, it is desired that the mesh be feature adaptive, so as to decrease the number of meshing elements of the overall mesh, yet be fine enough to capture all the necessary details and features in the domain. Kobbelt et al. [51] provide a framework to identify and extract features in the domain. Ho et al. [44] present a realtime feature editing tool.

2.2.6 Visualization Techniques

Different schemes are available for visualizing the model based on *volumetric representations*. Traditional techniques like *ray casting* [5, 103], *volume rendering* [36, 37], *iso-surface extraction* [40, 41] or *level set methods* [80, 82] can be used. For efficiency reasons, iso-surfaces are used for rendering volume sculpted models. First, each voxel of the object space is visited and the (polygonized) iso-surface is extracted and rendered. Then, for subsequent operations, as only certain regions of the model change during sculpting, an incremental marching cubes algorithm is used [41, 82]. Due to recent advances

in the graphics hardware, most of the (computationally expensive) polygon transformations, manipulations and rendering can be done at the hardware level on the graphics card of the display [36, 80], which results in significantly improved rendering rates and are almost real time and interactive. Elvins [35] has presented a good survey on volume visualization algorithms.

2.3 Computational Physics Modeling

Computational modeling deals with the simulation of actual physics of the model. For the purposes of this research, physically based computational modeling will be used in conjunction with the geometric domain models to perform realistic physics driven computer simulations. Much literature [34], [23] is available on studying the multi-scale physical models behind most problems. In such cases, there are usually two (or more) physical models describing the domain at different scales, e.g. crack propagation at the macro-scale and fracture analysis at the micro-scale. The key issue here is how to pass information and data from one model to the other and back. A large amount of literature is available that addresses these problems, chief among them being heterogeneous multi-scale methods [34], domain decomposition techniques [100], multigrid approaches [63] and wavelet based methods [73].

Selected work on the modeling efforts in *single physical scale*, yet large scale domain models is presented. Since this scope is very large, heat conduction problems are presented as these are related to the selected case study domain of SLS (See Section 2.4 below). Broadly speaking, the different models

are classified as follows.

2.3.1 Types of Domain Models

Different models have been proposed to model the conduction of heat across a domain. While some of them involve modeling the local *flow patterns* using the underlying physics of the problem, other approaches have relied on a more stringent (global) mathematical formulation and subsequently solving it on a grid that approximates the given domain. These are called *continuum models* and *discrete models* respectively. These are described below.

2.3.1.1 Continuum Models

Traditionally, this approach has involved the use of Finite Element (FE) methods to solve the diffusion problem. First a mesh is generated that closely approximates the domain of interest. Then, (partial) differential equations governing the physical process are defined on this mesh and solved numerically. This is a computationally intense approach, especially if the mesh sizes are very large (of the order of 10^6 mesh elements or higher) [9], [109].

2.3.1.2 Discrete Formulations

Often it is not possible to generate an exhaustive geometric model that captures all the intricate features of the domain. Furthermore, due to the limited computing resources available, it might not be feasible to numerically solve a differential equation on a dense mesh. In such cases, discrete formu-

lations are often preferred. Classical examples include *Monte Carlo* (MC) simulations, *Brownian Dynamics* (BD), and *Newtonian Molecular Dynamics* (MD) [97].

Compared to the discrete schemes, the continuum models yield a more accurate solution. The accuracy can be further improved by using higher order approximations in the numerical solver. In general, continuum models are the preferred choice for modeling large scale domains because of their *feature adaptivity*. Complicated geometries can be modeled in continuum finite element methods. This, along with adaptive timesteps can help simulate the diffusion process for longer timescales.

2.3.2 Multi-Scale Computational Modeling

In most multi-scale domain models, the coarse model (larger length scales) is modeled with continuum properties. The refined meshes (smaller length scales) are modeled typically with quantum mechanics, molecular dynamics or Monte Carlo simulations, depending upon both the actual magnitude of the length scale and the physical domain [97, 108]. In general, there are two "coupling" schemes for the information to be communicated between the coarse and refined meshes [108].

1. Hierarchical multi-scale coupling: Here, the information from one model is used to drive the simulation for the other model. In practice, at each time step, the coarser grid imposes a set of boundary conditions for the refined grids at the interface between the two. Note that the solution

can be obtained either by numerically solving the physical model, by analytical expressions for the refined mesh, or even by physical experimentation. Then, the coarser mesh uses this information to advance in its simulation.

2. Concurrent (or embedded) multi-scale coupling: This scheme allows both the coarse and refined solutions to concurrently co-exist in the same given region. The coupling between the two grids is handled by a traditional bridging scale method such as [34, 55, 78].

2.3.3 Heat Conduction Continuum Model

The basic Fourier heat conduction model for a 3D domain is given by Equation 2.3

$$K_x \frac{\partial^2 T}{\partial x^2} + K_y \frac{\partial^2 T}{\partial y^2} + K_z \frac{\partial^2 T}{\partial z^2} + q''' = \rho C_p \frac{\partial T}{\partial t} \tag{2.3}$$

where, $T = T(x, y, z, t)$ is the temperature distribution at any point (x, y, z) at any given time t in the domain Ω; K_x, K_y, K_z are the thermal conductivities of the material in the x, y, z directions; ρ is the material density; C_p is the specific heat of the material; and q''' is the rate of internal heat generated in the domain per unit volume.

In situations where the material can be assumed to be isotropic, $K_x = K_y = K_z = K$. Then, the above equation simplifies to

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{q'''}{K} = \frac{\rho C_p}{K}\frac{\partial T}{\partial t} = \frac{1}{\Psi}\frac{\partial T}{\partial t} \tag{2.4}$$

where Ψ is the thermal diffusivity of the material; the diffusivity is the ratio of the thermal conductivity to thermal capacity and it is a measure of the ability of the material to conduct heat with respect to time. Note that any effects due to phase changes, convection and radiation in the above model are neglected. (See Section 2.4 for details.)

A well posed problem of heat conduction over a given domain also requires a description of the boundary and initial conditions. For the example case of an 3D problem, the boundary conditions can be defined in the following ways:

- Dirichlet Boundary Conditions: Here, the value of the function is defined as $T(x, y, z) = F$. Thus, the actual function is known. This type is used to describe surfaces with (constant) known temperature profiles.

- Neumann Boundary Conditions: Here, the value of the function is defined as $\frac{\partial T}{\partial s}.\widehat{n} = F$, where $\widehat{n}$ is the surface normal. Thus, the derivative of the actual function is known and is used to describe the net heat flux over a surface.

- Robin Boundary Conditions: Here, the value of the function is defined as $\alpha T + \beta\frac{\partial T}{\partial s}.\widehat{n} = F$, where α and β are constants. This is a combination of the Dirichlet and Neumann conditions.

The initial conditions are given by usually describing the initial temperature distribution $T(x, y, z, 0) = T_0$.

2.4 Case Study Domain: Selective Laser Sintering

Selective Laser Sintering (SLS) is a rapid prototyping process. A layer of powder is selectively heated up to its fusing point by a focused laser beam. This causes the layer to bind to the previous powder layer. By manipulating the position of the laser beam, a 3D shape can be formed out of the powder. Complex shapes retaining functional properties can be formed by varying the materials and other process parameters [72].

The powder bed comprises both the powder particles and air. Thus the average composition of the bed is defined by the porosity P of the powder. This is given by Equation 2.5, where V is the volume of the powder and V_s is the volume of the entire sample.

$$P(\%) = 100(1 - \frac{V}{V_s}) \tag{2.5}$$

2.4.1 Heat Transfer Mechanisms for SLS

The heat transfer process in the SLS powder bed comprises the following different mechanisms. This is shown in Figure 2.1.

1. Thermal conduction between the solid powder particles
2. Thermal conduction between the trapped air particles

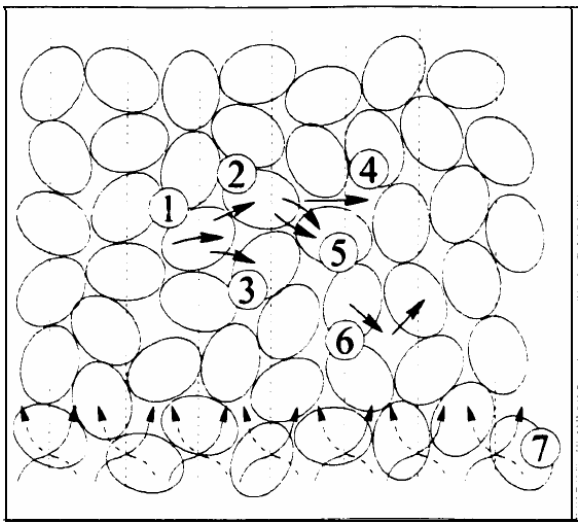

Figure 2.1: The different modes of heat transfer are shown in the powder bed here. [72]

3. Thermal radiation between the surfaces of the powder particles

4. Thermal radiation between the neighboring air particles (void regions)

5. Thermal conduction through the fluid film adherent to any surface

6. Thermal convection between the solid powder particles

7. Thermal convection due to the lateral mixing of the air particles

The total bulk thermal conductivity of the material is a function of all of the above heat transfer mechanisms. In reality, all these modes of heat transfer are coupled and it is difficult to model the total thermal conductivity of the powder bed. Usually, experiments are conducted to determine the net thermal conductivity of a powder bed for a prescribed working environment.

Furthermore, the thermal conductivity of the powder bed also depends upon the temperature of the powder and the geometric configuration of the bed. Detailed analysis of these different factors is presented by Norrell [72].

2.4.2 Modeling Heat Percolation

Many models have been proposed to understand the interesting and intricate phenomenon of heat conduction through porous media like the powder beds in SLS. The primary goal in this research is to first approximate the *effective thermal conductivity* for the powder bed.

While a lot of effort has focused on physical experimentation to understand this, analytical models have also been proposed. The work can broadly be divided based on purely analytical modeling and empirical modeling. The analytical approaches can be further categorized as based on continuum models or representative element volume based approaches [72].

2.4.2.1 Theoretical Models

Purely volume weighted schemes have been proposed as (preliminary) approximate models [15]. These models are simplified, in that they consider only the overall heat conduction without looking at the packing arrangements and the inter-particle and particle-air contact areas. Bear [15] and the Xue-Barlow [105] models are good examples for theoretical models.

Another approach for the net heat transfer phenomenon has focused on creating analogous electrical networks for the composite medium of powder and

air. Using the concept of electrical resistances to model thermal conductivity, an electrical network can be built for the entire domain. This is described in detail in [60] and [61].

Finite elements have also been used in modeling the heat flow through such graded composites [47]. Here, the entire domain consists of a discrete network of points (placed over the simple lattice structure). Then, all the bonds in this network corresponding to the conducting clusters are tagged and assigned a high conductance value. All the remaining bonds correspond to the non-conducting medium of the composite. Then, Kirchoff's laws [47] are applied over various elements to determine the net electrical current (flow) across the medium.

2.4.2.2 Representative Element Volume Based Models

The arrangement of the powder particles in the bed is assumed to be uniform everywhere in all directions. Then, analyzing a representative element is indicative of the entire bed. These are also called *unit cell based models*. They can be uniform cubic structures or orthorhombic structures, depending upon the orientation with respect to the principal axes. In all of these models, the powder particles are idealized as spheres.

The standard uniform cubic cell geometries focus on *simple cubic*, *body centered cubic* and *face centered cubic* structures. In these, fractions of the spherical powder particles are present at the 8 corners of the unit cell. Their geometries are fixed, hence their porosity values are also known [25]. This

does somewhat restrict their applicability to a general packing arrangement.

Krupiczka [52] has presented a few models that predict the thermal conductivities for these 3 geometries. The models provided by Yagi-Kunii [106] assume that the gas trapped between the spherical particles is motionless and hence, the effects of radiation and convection are neglected.

It is also worthwhile to mention the work by Chan-Tien [25] which agrees very closely with experimental results for powders with low thermal conductivities. However, the model predicts thermal conductance, while the research here focuses on the thermal conductivity.

2.4.2.3 Empirical Models

Empirical models have also been proposed on the basis of physical experimentation for certain types of powders over a range of working temperatures. While these models are usually restricted because of the specific powder types, they serve as a good validity check for the above models. Sun et al. [98] present such an empirical model, which is not material specific, but assumes that the thermal conductivities of both the particles and the air are low. This model has been developed almost exclusively for the selective laser sintering process.

2.5 Summary

A brief review of the relevant related work is presented in this chapter. Grid based geometric representations and volumetric sculpting are presented.

Computational physics modeling, in particular heat conduction in powder beds is introduced. Next, the first stage in the research pipeline, *geometric modeling*, is described in chapter 3.

Chapter 3

Geometric Modeling

In this chapter, the process to construct a comprehensive domain geometrical representation starting from the raw input data is described. Next, the scheme to *sculpt* together these volumes to construct a hierarchical adaptive volumetric representation of the complete domain is outlined.

3.1 Creating Volumetric Representations

The first step in the algorithm is to convert all of the input data into a common representation that can then be used for further processing. For this, the standard *volumetric representation* technique is chosen. In this, the object is represented by a (regular or adaptive) grid with a geometric function defined at each grid point. This function describes the shape and topology of the object. Tri-linear interpolation is used to compute the value of the function in the interior of the grid. Standard schemes are available to reconstruct the geometric object from a given volume: primal contouring a.k.a *marching cubes* [58], dual contouring [49], volumetric meshing [109] etc. to name a few. In this section, schemes for the construction of these volumetric representations from the different types of input data are presented.

If the input data is presented in the form of imaging data, then, the data is in the form of 2D slices with each *pixel* representing the intensity value in the datasets. Note that this physical interpretation is perfectly valid especially for data taken from conventional imaging modalities like MRI, CT and EM. Then, a series of initial pre-processing steps is performed to smooth noise and enhance sharp features: First, *contrast and image enhancement* are performed on this data to remove the noise and enhance the features in the dataset. This is followed by *image filtering* to further enhance the 1D and 2D features, followed by *image classification* whereby the different feature intensity intervals are identified and tagged. Next, *image segmentation* is performed on the data, where different regions of the dataset are grouped together based on their intensity values. This produces a set of (closed) contours which are then *tiled* together to reconstruct the original surface. The resulting surface can then be converted into the volumetric representation by using traditional distance fields [35]. Details about the imaging to volumetric representation schemes can be found in [12].

If the input data is presented in the form of molecular data, first, the data is parsed into a hierarchical and robust data structure called the *Flexible Chain Complex (FCC)* [8]. This is the internal representation for each molecule that is capable of storing the primary, secondary and tertiary structures of the protein. Next, each atom in the molecule is approximated by a Gaussian function which is defined at the center of the atom with its radius proportional to the Van Der Waals radius of the atom. Then, this function can

be simply sampled over a volumetric grid to compute the desired volumetric representation of the protein. Details about this scheme are presented in [8].

If the input data is presented in the form of CAD data, the data (a boundary representation of a manifold surface) is already present in the form of tessellated mesh elements (triangles or quads). In such cases, a grid can be simply defined over the desired domain and for each grid point, the signed distance to the surface can be computed. Note that this scheme is the basic routine that can be used for almost all types of input data, as almost all objects need to be tessellated before they can be rendered. Once an object is tessellated, this routine can be called to convert it into a volumetric representation. Details about this scheme are presented in Section 3.2.

Finally, if the input data is presented in the form of implicit surfaces, then the task becomes trivial. A grid can be simply defined over the desired region and the input data can be sampled over this grid. Note that this form of data includes the common geometric primitives sphere, cylinder, block and torus. All of these are treated internally as implicit functions and then converted into volumetric representations.

Clearly, most types of surface definitions can be processed. This is one advantage of choosing the volumetric representation as the common data representation type. Also, this ensures that when performing boolean operations on two objects only their respective function values defined at the same points in space need to be compared. Thus, the operations become very simple and computationally cost effective [41, 103].

In the remainder of the chapter, the different sub-problems in constructing the geometrical representations for a multi-scale domain are discussed in detail. The models for both the macro-scale and micro-scale geometries will be developed independently at their respective length scales. Then, the models for the intermediate scales will be generated and finally sculpted together for the comprehensive domain representation. Details of the application of this scheme to an example domain are presented in Section 3.7.

3.2 Signed Distance Field Computations

Computing accurate signed distance fields is a very crucial pre-processing step in the pipeline, while generating the volumetric models. As mentioned above, the brute force algorithm is very slow and often takes *days* to compute the complete distance fields for large resolution datasets (128^3 or higher). Hence, it is very important to achieve a speed-up using the latest state of the art schemes. Here, a brief overview is presented for the modifications that have been incorporated in the standard algorithm to achieve a very high speedup. The time to compute the distance fields is now reduced to a few *seconds*.

Both the space partitioning and the distance transforms schemes have been combined in the current approach to compute fast and accurate distance fields calculations (Please refer to Section 2.2.2). Starting from the input domain, first an adaptive octree[1] is constructed. This acts as the space par-

[1] An octree is a space subdivision technique where in 3D, each grid cell is uniformly divided into 8 equal cells. For more details, please refer to [37] and [90].

titioning data structure. This also provides a list of the boundary voxels and cells.

Then, the *sign* for each of the grid points is computed. This is a useful piece of information that is required by the distance propagation algorithm. A numerically stable, accurate and computationally inexpensive algorithm introduced by Segura et al. [90] for these computations is used for this. Next, the Euclidean distances for these boundary voxels are computed using the scheme given by Payne et al. [79]. Note that just this information is enough to extract the isosurface locally.

Next, these function values are propagated into the volume using the 3×3 chamfer city distance transforms[2] [84, 85]. This is a very fast algorithm that can compute the approximate values of the distance fields at the interior voxels. However, this is an approximate scheme and in reality, better results are achieved by storing at each grid point the closest point to the surface [102]. Propagating this information to the interior voxels using the distance transforms yields an accurate representation.

3.3 Scale Bridging

The sub-problem here is, given two geometries (volumetric representations in this case) at different length scales, providing a smooth transition from the macro-scale to the micro-scale. The data structures have to be hier-

[2]The SDF values for the new voxels are computed from its neighbors by using a distance template. In 3D, a $3 \times 3 \times 3$ neighborhood is used.

archical and scalable. Once the volumetric representations for S (referred to as V_s) and the C_i (referred to as V_c) are created, the next step is to register them together in one volume.

The actual difference in scales between the S and the C_i surfaces determines the nature and complexity of the data structures. For example, consider the case of the surface roughness example shown in Section 3.7, where S is of the order of a few *centimeters* (10^{-2}), and C_i are of the order of a few *microns* (10^{-6}). The difference in scales is *4*. This is typical of most applications, with some domains having even higher scale differences.

For a dense particle distribution, where most of the particles are scattered over the entire domain, it is possible to use a high resolution single volumetric representation that spans the entire length scales of the domain. But, for sparse distributions, having a large single representation results in unnecessary memory and processing overheads that lead to exponentially slower simulation times (See Section 5.5 for a detailed analysis of the time and cost analysis for simulations based on volumetric representations.). Hence, intermediate volumes are needed to bridge the scales, thus providing a smooth transition from the macro-scale to the micro-scale.

Consider the volume V_s. From the particle distribution function F, the exact location of the particles in this volume is known. Hence, a recursively adaptive subdivision of this volume is performed, such that the locations of C_i have a finer geometric representation. The goal of this step is to bridge the scale difference between the volumetric representations for the different

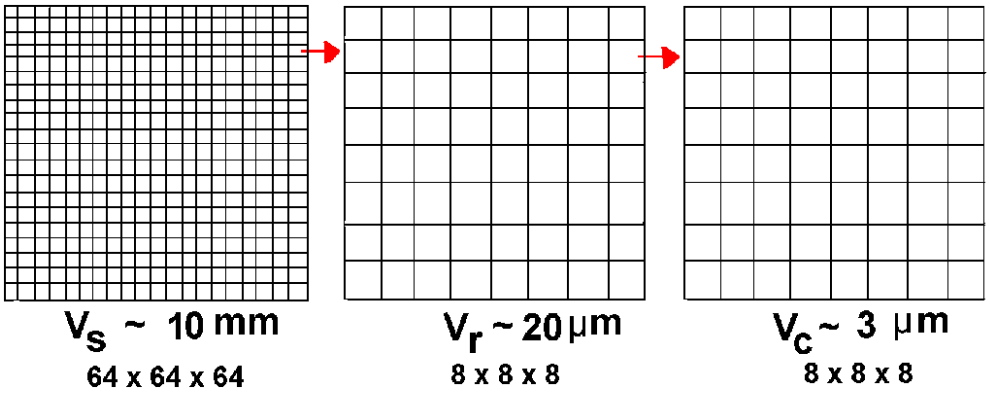

Figure 3.1: The sketch of the hierarchical data structures used for scale bridging.

components of the domain.

In the example shown in Figure 3.1, the resolution of V_s is chosen to be of $64 \times 64 \times 64$[3] (for the sake of clarity, only a 20×20 grid for V_s is shown.). The resolution of the V_c is chosen to be $8 \times 8 \times 8$, because there are relatively few complex features in these particles. Now, each cell in the volume V_s that contains a particle C_i is further subdivided into an uniform intermediate volume V_r (The sub-volumes in Figure 3.1 are shown with 8×8.). Then, depending upon the exact location of the particle C_i given by F, the two volumes can be *sculpted* together, which are now of the same scale. This is explained in Figure 3.1. The smallest resolution (grid cell size) of V_s is $\frac{10}{64}$ mm, and that in the V_r is $\frac{10}{64} \times \frac{1}{8}$ mm or approximately 20 μm and finally in V_c is $\frac{10}{64} \times \frac{1}{8} \times \frac{1}{8}$ mm or approximately 3 μm. Thus, we can successfully create

[3]For reasons mentioned in Section 5.4, the chosen resolutions are 2^m, where m is an integer between 2 and 8.

a smooth transition from centimeter scale to the micron scale.

Note that the choice of the resolution of V_r is affected by the actual difference in scales. In turn, it affects the level of hierarchy of the volumes (i.e. the number of intermediate V_r). If the resolution of V_s is chosen as $64 \times 64 \times 64$, then, V_r needs to be $8 \times 8 \times 8$ to take us into the micrometer domain range. Finally, each cell in the last V_r can have a pointer to the actual volume V_c, whose resolution is also $8 \times 8 \times 8$.

Thus, by creating a series of intermediate volumes V_r, it is possible to bridge the scale difference between V_s and V_c. In practice, a maximum resolution of $256 \times 256 \times 256$ is recommended because of dynamic memory limitations and this leads to hierarchical V_r volumes and a minimum limit of $8 \times 8 \times 8$ is chosen to have fewer V_r subvolumes.

3.4 Generating the Intermediate Volumes

The sub-problem here is, given a volume representation at a fixed resolution V_o, generating a high resolution volume V_f over the same domain space. Both the volumes need to be co-registered so that the transition from one resolution to the other is smooth.

There are two approaches to do this computation. In the brute force approach the same techniques mentioned in Section 3.2 can be used to generate a fixed resolution volume using the signed distance fields. For this, the desired volume V_f can be treated as an individual volume and generated from the input

domain description described in Section 3.1 above. While this guarantees a good and accurate volume, the major drawback is the need to re-create the computationally expensive octree for the input description. Also, once the volume is computed, to form a conforming and co-registered volume, an interpolation might have to be performed between the newly computed V_f and the given V_o. This is an approximation and does not work well in practice. Hence, an alternate approach is proposed.

Given the volume V_o, it is possible to define a tri-linear function approximation over the entire domain using Equation 2.1. This gives an approximation to the true function based on the sampling rate of V_o. Then a grid of the desired resolution can be defined anywhere in the domain and the V_f volume can be generated by simply sampling this function at the grid points. Note that this approach guarantees that the two volumes are co-registered, while avoiding any costly computations.

3.5 Volumetric Sculpting

Here, the sub-problem is given two (or more) overlapping volumes V_v, *sculpting* them together to yield a final volume V_d such that V_d is the boolean union of V_v. First all the volumes V_v have to be sampled at the same exact locations in space (i.e. they are overlapping). This can be done by simply using the tri-linear interplant defined in Equation 2.1 and sampling it at the desired locations on a fixed grid. Once all the volumes V_v are overlapping, then sculpting can be performed by simply using a series of *if-else* branching

statements to determine the final function value at each grid point. This is shown in Figure 3.2.

3.6 Volumetric Domain Representations

Volumetric grids can be interpreted and leveraged in the following manner:

- Domain decomposition: Volumetric representations provide a natural framework to subdivide the domain into smaller regions. For a regular grid with no embedded subvolumes, the grid can also double as a volumetric mesh comprising hexahedral elements, where each grid cell is one hexahedron. A very similar scheme is used in [9] to extract hexahedral meshes using distance fields.

- Hierarchy: The multi-scale domain volumetric representation described in Sections 3.3 and 3.4 provides a hierarchy of subvolumes. This can be visualized as a tree structure, with the "root" node as V_s. The embedded subvolumes V_r can then be expanded and thus the tree can be traversed to reach the final nodes V_c.

- Adaptivity: The embedded subvolumes are added to V_s only when higher resolution is needed at a certain portion of the domain. Thus, depending upon the situation, subvolumes could be added to further refine a geometric feature, as described in Section 3.3. However, for numerical

#define abs(val) ((val) < 0 ? -(val) : (val))

input: overlapping volumetric representations V_1 and V_2
output: sculpted volumetric representation V_f, where $V_f = V_1 \oplus V_2$

for each midpoint *P(i,j,k)* in the volumes
{
    v1 $\leftarrow V_1(P)$
    v2 $\leftarrow V_2(P)$
    if ((sign(v1) == 0) AND (sign(v2) == 0))
        sign(v3) $\leftarrow$ 0
    else if((sign(v1) == 0) AND (sign(v2) < 0))
        sign(v3) $\leftarrow$ -1
    else if((sign(v1) == 0) AND (sign(v2) > 0))
        sign(v3) $\leftarrow$ 0
    else if((sign(v1) < 0) AND (sign(v2) == 0))
        sign(v3) $\leftarrow$ -1
    else if((sign(v1) > 0) AND (sign(v2) == 0))
        sign(v3) $\leftarrow$ 0
    else if((sign(v1) > 0) AND (sign(v2) > 0))
        sign(v3) $\leftarrow$ 1
    else if((sign(v1) > 0) AND (sign(v2) < 0))
        sign(v3) $\leftarrow$ -1
    else if((sign(v1) < 0) AND (sign(v2) > 0))
        sign(v3) $\leftarrow$ -1
    else if((sign(v1) < 0) AND (sign(v2) < 0))
        sign(v3) $\leftarrow$ -1
    end if
    value(v3) $\leftarrow$ min(abs(v1), abs(v2))
    $V_f(P) \leftarrow$(sign(v3) $\times$ value(v3))
}
end for

Figure 3.2: Algorithm for the volumetric sculpting routine

simulation, certain regions of the domain might also need to be examined at a higher resolution. It is also possible to create subvolumes for such regions by using any of the methods described in Section 3.4 and skipping the final volumetric sculpting step.

- Overlaid grids: While, the subvolumes created in the scale bridging operation are *sculpted* together into the parent volume, they can also be thought of as individual volumes. This idea is crucial in the numerical simulation algorithm, where each subvolume can then be numerically solved separately (For details, see Sections 4.4 and 4.5).

- Adaptive mesh refinement: The resulting grid structure after volumetric sculpting looks very similar to the Adaptive Mesh Refinement (AMR) structure proposed by Oliger and Berger [18]. In their algorithm [18, 77], an error estimation procedure based on the numerical solution determines any additional refinements on the grid. In contrast, in the present case, initially, the geometry of the domain determines the refinements. However, for subsequent refinements, an error estimation procedure can also be used to produce a smoother function.

3.7 Surface Roughness Models

Consider an example of constructing a geometrical model for surface roughness (surface irregularities or imperfections) of idealized spherical particles. These problems impact the areas of tribology and fracture mechanics,

where a wealth of research has focused [1, 6, 7, 61]. The problem is essentially multi-scale in nature, where at one end of the scale, the bulk material is present and at the other end, the microscopic surface irregularities are present. The length scales vary from a few millimeters to a few microns across the entire domain [7]. The existing research focuses on the aspects of contact resistances at the micro-scales and their effects at the macro-scales.

For the example of surface roughness models, implicit surface representations for perfect spherical shapes are first formulated and then converted into volumetric representations using the scheme outlined above. Then, *noise* is introduced into these models using small perturbations along the boundary voxels. This is achieved by simply tweaking the signed distance values within a certain user defined tolerance σ to locally change the shape of the resulting iso-surface. Bahrami et al. [6, 7] show that at the micro-scale, the surface roughness models are defined by their distribution over the idealized surface, Root Mean Square (RMS) values of the perturbations and the mean absolute surface slope values. These three parameters control the distribution, size and shape of the perturbations. In the case of volumetric representations, a space subdivision is provided naturally by the octree structure. Since, the perturbations are only over the boundary of the idealized geometry (spheres), all the boundary voxels of the volumetric representations are first selected. From this list, the set of boundary voxels that are close to the contact areas of the idealized spheres are tagged. These are the basic *units* for further processing.

The signed distance values at the grid points of these tagged voxels are

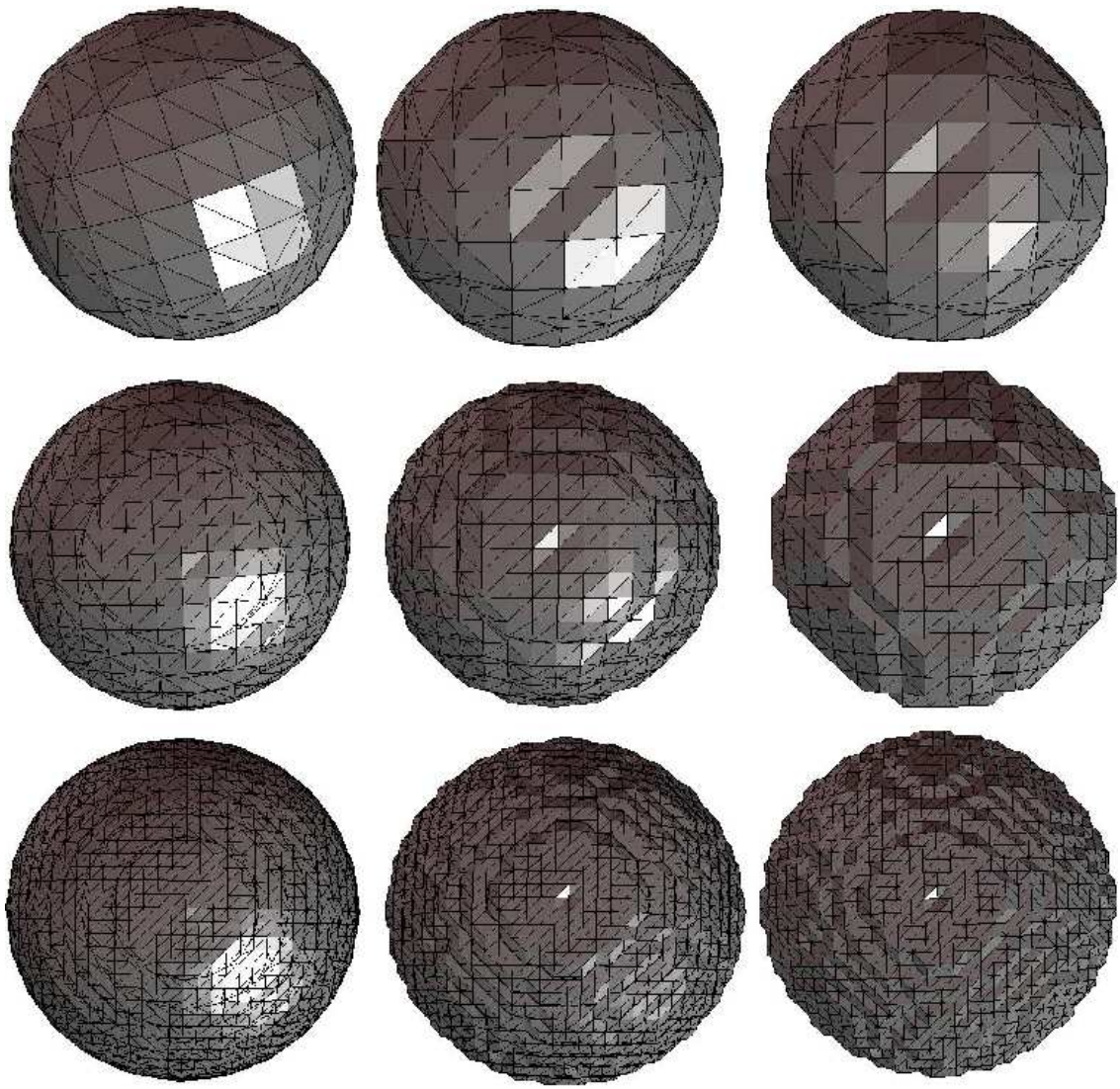

Figure 3.3: Surface roughness models with different values of σ and grid resolution.

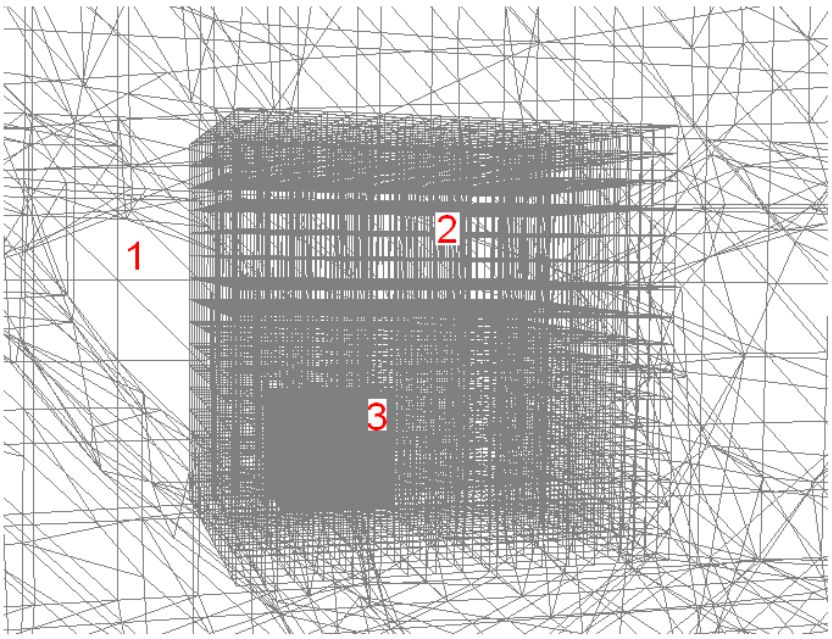

Figure 3.4: Comprehensive geometric domain representation showing 3 levels of hierarchy.

then perturbed by adding a small amount σ equal to the desired RMS value of the perturbations. This controls the size of the roughness models. This is shown in Figure 3.3. The models shown in the first column are the idealized spherical geometries. The second column shows the models with $\sigma = 0.04\mu m$ and the last column shows the models with $\sigma = 1.0\mu m$. All models shown in Figure 3.3 are obtained from an unit sphere.

The shape of the models can be controlled by varying the grid resolution of the volumetric representations. Figure 3.3 shows the different shapes obtained by different grid resolutions. The first row is of resolution $16 \times 16 \times 16$,

the center row is of $32 \times 32 \times 32$ and the bottom row is of $64 \times 64 \times 64$.

The intermediate volumetric representations are generated by simply computing high-resolution signed distance functions using the same set of input implicit surface definitions. This guarantees that in the limit, as the grid spacing decreases, perfectly smooth spherical shapes are generated.

The comprehensive domain representation is shown in Figure 3.4 where the macro-scale volumetric representation grid has a resolution of $64 \times 64 \times 64$. The intermediate and micro-scale volumetric representation grids are each of $8 \times 8 \times 8$ resolution. The volumetric grids labelled 1, 2 and 3 are the V_s, V_r and V_c respectively. With the parameters chosen above, the smallest grid cell size in V_c is $3\mu m$. Thus the smallest feature that can be efficiently modeled with this representation is of the order of a few μm.

3.8 Summary

As described in this chapter, starting from the surface descriptions for S and C_i, a comprehensive (geometric) domain model using volumetric representations is first constructed. The grid that is constructed is re-used to create the initial scaffolding of the domain, and then a set of functions are described over this grid. In this section, it can be seen that using geometric shape functions such as the signed distance function, it is possible to create efficient and accurate descriptions of the domain geometry. In the next chapter, this framework is used to describe the computational function(s) used for numerical simulations.

Chapter 4

Computational Modeling

This chapter presents the approach for computational modeling. As mentioned in Section 2.1, the scope of the function models is restricted to domains which have one set of underlying physical equations. This assumption can be relaxed without affecting the geometric modeling aspects of the research. *Any number of functions* can be defined over the initial scaffold, e.g. some to model the macro-scale physics and the others to model the micro-scale physics. However, these models will then have to communicate information and data back and forth for each timestep in the simulation. To avoid these costly computations, only *one* function is needed to describe the physical model of the domain. This greatly simplifies the algorithm for the numerical solver.

Once the volumetric representation for the domain is constructed, the computational model for the domain is then developed. Next, the numerical approximation of the solution is formulated. After the boundary and initial conditions are defined over the domain, a numerical simulation of the physical process is performed. In this case, the solution function is sampled at the grid points of the volume, similar to the geometric function defined in Section

3.1. Details of the application of this scheme to an example heat conduction problem are presented in Section 4.6.

4.1 Finite Difference Models

The scope of the computational modeling efforts is restricted by focusing on models applicable to the selected application domain of selective laser sintering described in Section 5.2. A simple heat conduction model is used to demonstrate the framework. Since an explicit solver[1] (for reasons explained in Section 4.5) is used, the necessary finite difference equations are provided here. The 3D heat conduction equation for an isotropic domain is given in Equation 2.4. For the case of no internal heat generation, this reduces to Equation 4.1.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = \frac{1}{\Psi}\frac{\partial T}{\partial t} \tag{4.1}$$

Consider the 3D neighborhood of a grid point P. The grid points incident to it are: $A^{'}$ and $B^{'}$ in the X direction, $C^{'}$ and $D^{'}$ in the Y direction and $E^{'}$ and $F^{'}$ in the Z direction. Assume that the boundary surface intersects the grid at points A, B, C, D, E and F as shown in Figure 4.1. These boundary points A, B, C, D, E and F are at distances $\alpha\Delta x$, $\beta\Delta x$, $\gamma\Delta y$, $\delta\Delta y$, $\lambda\Delta z$, $\mu\Delta z$, respectively from P. If the point P is a boundary grid point (as shown in the Figure 4.1), then all these incident edges are sign change edges[2] (Please refer

[1] In an explicit scheme, the new solution function values T^{n+1} are defined in terms of already known T^n function values.

[2] In reality, this situation can happen only in highly degenerate cases, where the material

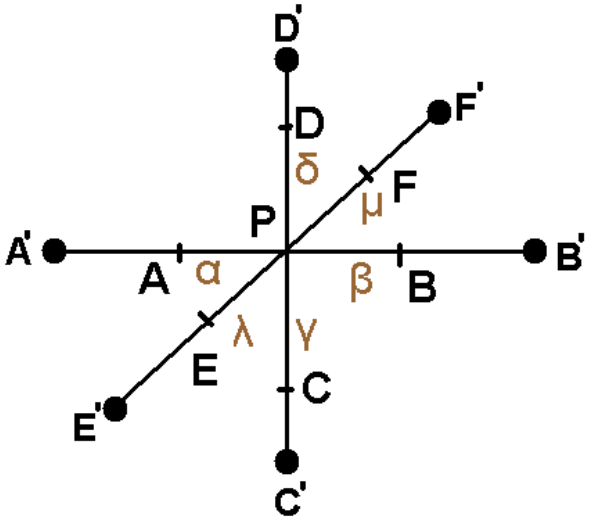

Figure 4.1: Finite difference scheme at the boundary.

to Section 4.3 for details). An explicit scheme based on *Forward Time Center Space* (FTCS) is used for transient state equations. Then the finite difference formulation for the Equation 4.1 is given by Equation 4.2.

$$T^{n+1}(P) = \frac{2r_x}{(\alpha+\beta)}(\frac{T^n(A)}{\alpha} + \frac{T^n(B)}{\beta}) + \frac{2r_y}{(\gamma+\delta)}(\frac{T^n(C)}{\gamma} + \frac{T^n(D)}{\delta}) + \frac{2r_z}{(\lambda+\mu)}(\frac{T^n(E)}{\lambda} + \frac{T^n(F)}{\mu}) + [1 - \frac{2r_x}{\alpha\beta} - \frac{2r_y}{\gamma\delta} - \frac{2r_z}{\lambda\mu}]T^n(P) \quad (4.2)$$

where $r_x = \frac{\Psi\Delta t}{(\Delta x)^2}, r_y = \frac{\Psi\Delta t}{(\Delta y)^2}, r_z = \frac{\Psi\Delta t}{(\Delta z)^2}$. The r_x, r_y and r_z are the Fourier numbers in the X, Y and Z directions respectively. This equation is the general form of the well known 7 point stencil for 3D Laplace equations. The derivation

at point P is *isolated* from its neighbors. Usually, for boundary points, the number of sign change edges is between 1 and 5.

for this is shown in Appendix A. The first 3 terms on the right hand side arise from the finite difference equation in each of the 3 spatial co-ordinate directions, while the last term is due to the finite difference in the temporal dimension from Equation 4.1.

For the case of steady state analysis, in the Equation 4.1 the right hand side vanishes. Hence, the corresponding finite difference model is given by Equation 4.3. Appendix A shows the derivation for this equation.

$$[\frac{1}{\alpha\beta(\Delta x)^2} + \frac{1}{\gamma\delta(\Delta y)^2} + \frac{1}{\lambda\mu(\Delta z)^2}]T(P) = \frac{1}{(\alpha+\beta)(\Delta x)^2}(\frac{T(A)}{\alpha} + \frac{T(B)}{\beta}) + \frac{1}{(\gamma+\delta)(\Delta y)^2}(\frac{T(C)}{\gamma} + \frac{T(D)}{\delta}) + \frac{1}{(\lambda+\mu)(\Delta z)^2}(\frac{T(E)}{\lambda} + \frac{T(F)}{\mu}) \quad (4.3)$$

4.2 Stability Analysis

For a numerical scheme to be stable, the error accumulated at any given timestep (due to various errors like truncation error, round-off error etc.) should be bounded and not propagated to the next timestep. It can be shown that for schemes to be stable, the Fourier number has to be less than unity [4]. For schemes with Fourier number larger than unity, the solutions are unstable and unreliable because the amplitude of the errors increases very rapidly from one timestep to the next.

The term *mesh ratio* m is defined as the ratio of time step to space

Table 4.1: Stability Analysis for constant mesh ratio m

Mesh Number	Mesh size (Δx)	Time step (Δt)	Fourier Number (r)
V_s	$\Delta x_1 = \sim 10^{-3}$	t_1	$\frac{t_1}{\Delta x_1}$
V_r	$\sim 10^{-5}$	$\frac{t_1}{10^2}$	$10^2(\frac{t_1}{\Delta x_1})$
...	...	...	...
V_c	$\sim 10^{-9}$	$\frac{t_1}{10^6}$	$10^6(\frac{t_1}{\Delta x_1})$

(mesh) step on all grids [34]. In their numerical simulations for hyperbolic equations, Berger et al. [18] and Almgren et al. [4] maintain a constant mesh ratio over both the spatial and temporal domains.

Choosing a constant mesh ratio m has first been experimented with. For the problem shown in Equation 4.1, Table 4.1 shows the results for this. By using a fixed mesh ratio and an initial timestep for the coarsest grid (V_s), the timesteps for the remaining grids are computed. Closer analysis reveals that the Fourier number for progressive grids keeps increasing in the same ratio as the mesh (or time) step. This means that as the grid is further refined, the Fourier number increases. This leads to unstable solutions. Hence, a constant mesh ratio cannot be chosen for the solutions.

However, if a constant Fourier number is chosen the timesteps for all grid sizes can also be similarly computed as above. This is shown in Table 4.2 for the problem in Equation 4.1. This ensures that the scheme is stable. Thus, constant Fourier numbers are chosen depending upon the dynamics of the actual domain that are modeled.

D'Acunto presents the theoretical proofs for the convergence and consistency of this explicit scheme for the heat conduction equation shown in

Table 4.2: Stability Analysis for constant Fourier number r

Mesh Number	Mesh size (Δx)	Time step (Δt)	Mesh Ratio (m)
V_s	$\Delta x_1 = \sim 10^{-3}$	t_1	$\frac{t_1}{\Delta x_1}$
V_r	$\sim 10^{-5}$	$\frac{t_1}{10^4}$	$\frac{1}{10^2}\frac{t_1}{\Delta x_1}$
...	...	...	...
V_c	$\sim 10^{-9}$	$\frac{t_1}{10^{12}}$	$\frac{1}{10^6}\frac{t_1}{\Delta x}$

Equation 4.1. Although the scheme he considered is for a simple 1D heat conduction, it is straightforward to extend it to the general case of 3D.

4.3 Modeling Initial and Boundary Conditions

The initial conditions are defined over the interior of the entire domain. The given function is simply sampled at the grid points of the volumetric representation. Within each grid cell, the same tri-linear interpolation given by Equation 2.1 that is used for the interpolation of the geometric function, is utilized.

For the boundary conditions, first, the exact points where the boundary of the input geometry cuts the grid need to be computed. Closer inspection of the geometry function (described in Section 3.2) reveals that evaluating the implicit function given in Equation 2.1 at an iso-value of 0, provides us with the required points along the edges of the grid (along with their connectivity information). Hence, all the *sign change edges*[3] in the volumetric

[3]An edge is called a sign change edge if one vertex lies inside (or on) the boundary of the surface and the other vertex lies outside (or on) the boundary. Thus the boundary is guaranteed to intersect the edge at a unique point.

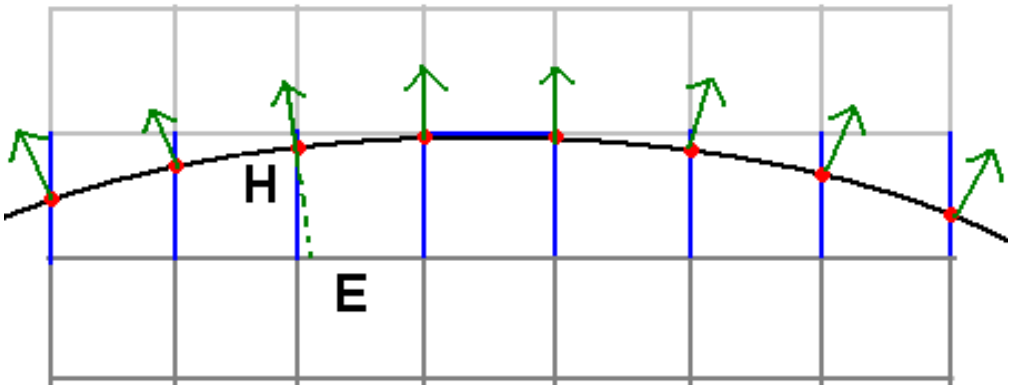

Figure 4.2: Neumann boundary conditions on a curved boundary. The sign change edges are shown in blue and the actual points of intersection in red. The surface normals at these points are computed and shown in green. At point H, the normal is flipped and intersected with the grid at point E.

representations are processed first, using the signed distance function. For each such edge, the exact location where the boundary surface intersects the edge is computed. Next, the surface normal at that point can be computed from the input surface geometry description. This information is stored at the grid vertex which is *outside* the boundary. This is shown in Figure 4.2.

The Dirichlet boundary conditions are the simplest to model as the function is explicitly given. In such cases, the function at the grid points of the volume is simply sampled.

The Neumann boundary conditions can also be similarly modeled with a slight modification. For the Neumann boundary condition at point H on the boundary, the intersection E between the surface normal at H and the grid *inside* the boundary is computed. For this, the normal needs to be flipped to direct it inside the boundary. Thus the Neumann condition at H is defined as

(see Figure 4.2):

$$\frac{dT}{dS}_{|H} = \frac{T(E) - T(H)}{d(EH)} \tag{4.4}$$

where $d(EH)$ is the Euclidean distance between points E and H.

For the sake of completeness, in the case of inhomogeneous materials (materials with different physical properties), the initial conditions on either side of the material boundary can be different. In such cases, at the *outside* boundary grid points, both the initial condition and the boundary condition are stored. For datasets with only one material, only the boundary condition at these *outside* boundary grid points is stored.

Once the initial and boundary conditions are imposed on the volumetric representation grids, the excellent theoretical work from the field of numerical simulations using AMR data structures can be leveraged. This is described in detail in the remainder of this chapter.

4.4 Steady State Simulations

In the domain geometrical representation, the coarse and fine grids are sculpted together to form one comprehensive representation. These individual sub-grids can also be considered independent grids, except at the boundaries, where they interface with the neighboring grids. It is this property of the representation that is exploited in the numerical simulations. The solver is set

up such that it can numerically march through each grid subject to the local boundary and initial conditions of the grid. At the end of each iteration, the function values across the common interface need to be synchronized.

In each iteration, the 7 point stencil for finite differences as given in Equation 4.3 is used for updating the values of the physical function during each iteration. After all the grid points are updated once, the embedded subvolumes are next updated. For each subvolume, the same 7 point stencil given in Equation 4.3 is applied to updates its grid points. However, a crucial step before doing this is to define the boundary conditions at the interface between the coarse and fine grids. This is essential in order to synchronize the function values from both the grids.

4.4.1 Boundary Conditions at the Coarse-Fine Grid Interface

For the embedded fine grid, the values at the interface with the coarse grid need to be specified before advancing it. This is done by linearly interpolating in space the values from the coarse grid. In the 3D grid representation, only the 8 corner grid values are known from the coarse grid cell. These values are used to interpolate the values along the 6 faces of the fine grid (the coarse-fine grid interface) using bi-linear interpolation.

4.5 Transient State Simulations

In general, there are two types of time dependant solvers: explicit solvers and implicit solvers. Depending upon the class of PDE, for stabil-

ity and convergence of the numerical solution, explicit solvers are used if the problems are non-stiff, and implict methods are used for stiff problems. For adaptively refined grids, time dependant solvers can also be classified as: synchronized time step methods and local time step methods [17, 54].

At each timestep, the macro-scale grid is first advanced. Next, the underlying micro-scale grids are advanced to the same time, using finer timesteps as given by the mesh ratio. Then, the continuity constraints at the interface of the micro-scale and macro-scale grids are imposed by interpolating the values from the two simulations. The details of this *synchronization* step using the *flux correction step* are given by Berger et al. [18] for 2D grids and generalized to 3D grids in [4].

Almgren et al. [4] provide an excellent review of the techniques used for time simulations on Adaptive Mesh Refinement (AMR) grids. In general, they have overlapping grids where for the overlapping portions, the coarse grids define the Dirichlet boundary conditions for the fine grids. Then, after both grids are advanced to the same time instant, the fine grids simply overwrite the values in the coarse grids at these overlapping regions. This is called the *grid restriction* step.

If the refinement level between two nested grids is r, then, for an explicit solver, for each coarser grid time step, the finer grid has to be advanced by r time steps. This is to ensure that the CourantFriedrichsLewy condition (*CFL condition*) number remains the same for all grids [54] to ensure stability and convergence.

4.5.1 Boundary Conditions at the Coarse-Fine Grid Interface

Linear interpolation in both space and time is used to compute the function values at the interface between the coarse and fine grids. However, in the case of the transient solver, this is complicated as the timesteps for integration are different for both grids.

Similar to the steady state solver, linear interpolation is used to compute the function values along the 6 faces of the interface. However, for the intermediate timestep values of the fine grid, for which the coarse grid function values at the 8 corner vertices are not defined, linear interpolation in time is used to compute the 8 function values. Once this is done, these values are used to linearly interpolate in space to compute the values along the 6 faces of the coarse-fine grid interface [17].

4.6 Heat Conduction Models

Consider an example of simulating the heat conduction process through a domain such as a metal bar. Consider the case of uni-directional heat flow where four faces of the bar are insulated and heat can flow only across the remaining two faces. Assume there is no internal heat generation and heat transfer in the domain results from a net temperature differential across the two un-insulated faces. The governing physics model for this process is given in Equation 4.1. The finite difference formulation for this is derived in Appendix A and implemented in the numerical solver for both the steady state and transient state models. The same process is modeled using a commercial third

party solver ANSYS [48] and the results are compared to the current solver.

4.6.1 Steady State Simulations

The steady state model for the domain is presented in Equation 4.3. The geometrical model for the domain as created in ANSYS is scaled 1:1:4 in the 3 co-ordinate directions and contains 108 hexahedral mesh elements. The grid spacing is the same in all 3 directions. The original geometrical representations are shown in the first row of Figure 4.3. The model created using volumetric distance fields has a grid resolution of 16. Both the geometric models have the same number of hexahedral mesh elements. Heat is allowed to flow only along the Z direction and these two faces are maintained at 100^oC and 500^oC, while the other four faces of this mesh are insulated. Initially, the temperature throughout the domain is maintained at 100^oC. The temperature distribution profiles are shown in the second row of Figure 4.3. The temperature distribution and the heat flux values across the domain are in close agreement with the results from ANSYS.

4.6.2 Transient State Simulations

The transient state model for the domain is presented is Equation 4.2. The necessary simulation parameters are shown in Table 4.3. For the numerical stability of the solver, the restriction on the time step is given in Equation 4.5.

$$\frac{\Delta t}{(\Delta x)^2} + \frac{\Delta t}{(\Delta y)^2} + \frac{\Delta t}{(\Delta z)^2} \leq \frac{1}{8} \tag{4.5}$$

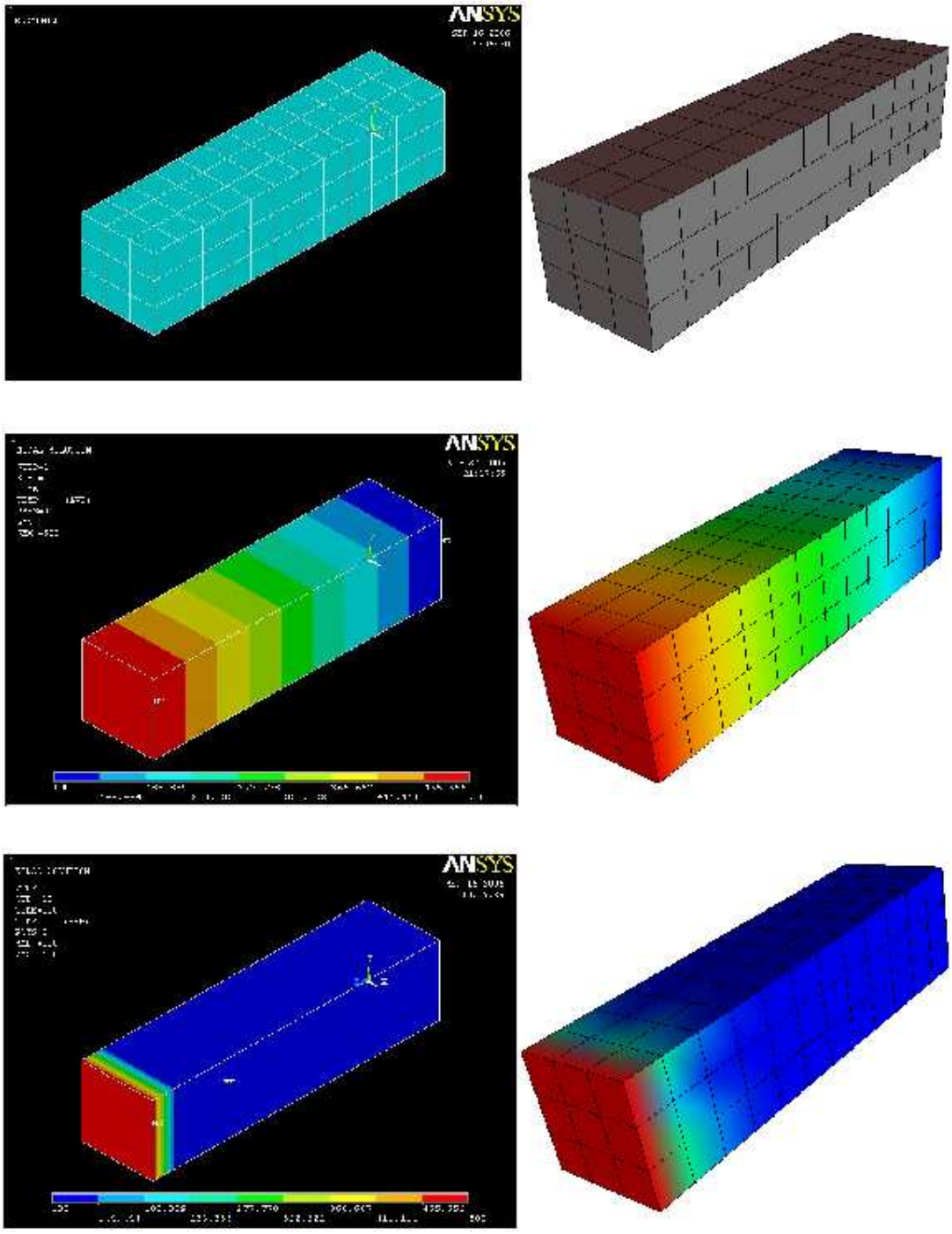

Figure 4.3: Results from the heat conduction simulation.

Table 4.3: Simulation values for 3D heat conduction simulation

Parameter	Symbol	Value
Grid spacings	$\Delta x, \Delta y, \Delta z$	1m
Thermal conductivity	K	5 W/m-K
Material density	ρ	920 kg/m^3
Specific heat	C_p	2.04 KJ/Kg-K
Start time	t_0	0 sec
End time	t_f	2 sec
Time step	Δt	0.02 sec

Using the time step value from Equation 4.5 for the chosen grid spacings as shown in Table 4.3, the temperature distribution profiles for the transient analysis after a simulation time of 2 seconds, are shown in the last row of Figure 4.3. These are similar to the ones from ANSYS at all the grid points in the domain. Differences in the interpolation scheme lead to small variances for temperature distributions and fluxes across the element shapes. Similar trends were observed for other values of simulation time.

4.7 Summary

The computational modeling work is presented in this chapter. An example parabolic heat conduction equation is considered and its numerical approximation for the volumetric representation framework is developed. The numerical solver based on this scheme is then implemented and the results from the simulation are validated with a commercial solver. Next, this framework is applied to the application domain of SLS in chapter 5.

Chapter 5

Application Domain: Selective Laser Sintering

In this chapter, the proposed multi-scale domain modeling framework is applied to the Selective Laser Sintering (SLS) process. Of special interest is the modeling of heat transfer processes through the powder bed.

5.1 Introduction

The powder bed for the SLS process consists of many layers of powder particles. The sintering process occurs when a laser focuses a beam of energy on these layers of powder. To efficiently model the heat transfer process through the powder bed, an approximate Effective Thermal Conductivity (ETC) for the powder is required. This is a function of several factors including the material properties of the powder and air and the geometrical arrangement of the powder particles.

The first assumption about the powder bed is that the powder packing is regular and powder particle size is uniform. Furthermore each particle is modeled as an idealized sphere. For the preliminary models shown in Section 5.2, a perfect spherical shape is assumed and then imperfections are introduced due to surface roughness for the later models in Section 5.3. Finally, the air

trapped between the particles is assumed to be stationary, so there is negligible convection between the air voids.

For the unit cell models, a key consideration is the contact areas of the powder particles. For simply touching spheres these are simply points. For the case of intersecting spheres, these are circles of finite radii. In the geometric modeling phase, the entire unit cell model is chosen as the macro-scale geometry and these contact areas are the micro-scale geometry where the grids are refined. Also, an uni-directional heat flow across two opposite faces of the cube is assumed. The temperature at the top face is maintained constant at 373^oK, while the bottom face is maintained at 323^oK. The initial temperature everywhere is also 323^oK. The other 4 faces of the cube are insulated, so that no heat can travel across them. These conditions are identical to the ones mentioned in the literature for polycarbonate-air beds [72, 98, 105] and are hence chosen because of their ease for comparing the results to theirs.

As described in Section 2.4, the packing arrangements for the Simple Cubic (SC), Body Centered Cubic (BCC) and Face Centered Cubic (FCC) structures are analyzed. These are the standard types of arrangements which are used to approximate most regular packing structures. The geometry and hence the porosity values for these structures are also fixed [7, 25]. While the SC structure gives an upper bound on the porosity, the FCC gives a lower bound on the porosity values for the packing patterns.

Table 5.1: Results for ETC (W/m-K) for Polycarbonate-Air bed

Geom	Continuum Models		Unit Cell Models		Empr	Multi-Scale	
Type	Bear	Xue-Barlow	Krupiczka	Yagi	Xue	Sphere	SR
SC	0.108	0.095	0.121	0.073	0.072	0.0912	0.0914
BCC	0.131	0.120	0.121	0.107	0.106	0.1161	0.1166
FCC	0.141	0.130	0.121	0.120	0.121	0.1469	0.1480

5.2 Unit Cell Models

For a preliminary study of the SLS domain, just the multi-scale domain of idealized spherical particles was considered. Table 5.1 shows the simulation results for a powder bed comprising polycarbonate-air particles. Polycarbonate is widely used in the fabrication of design prototypes [72] and is a good material for comparing the present models with the theoretical models shown here. The thermal conductivities of bulk polycarbonate and air are 0.18 and 0.028 W/m-K respectively.

The results were compared with the available models in the literature outlined in Section 2.4.2. Some of these models are also shown graphically in Norrell's dissertation [72] in Figure 2.2 and the general trends predicted from the simulations agree with those models. As the porosity of the powder bed decreases, the particles are more closely and tightly packed together. This provides more conduction paths for heat transfer, so the net thermal conductivity of the bed increases. At the working temperature ranges that were chosen, the effects of radiation are negligible [72].

An example for the BC dataset is shown in Figure 5.1. Only two contact regions are shown with multi-scale volumetric representations. The

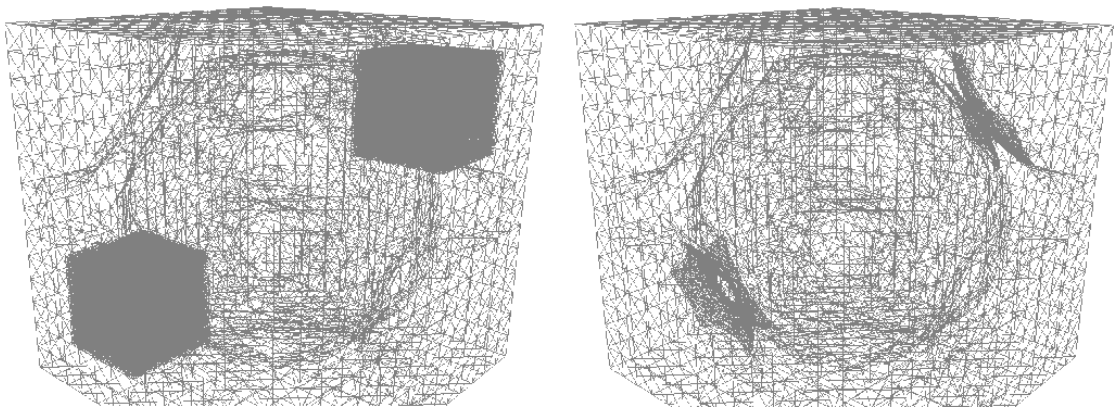

Figure 5.1: The multi-scale domain model for the body centered cubic dataset. Two contact regions are shown with multi-scale representations.

grid to the left shows the entire multi-scale domain representation, while the grid to the right shows the iso-surfaces from the same representation. Note the high resolution surfaces from the subvolumes at the contact areas. The macro-scale grid has a resolution of $16 \times 16 \times 16$, while the subvolumes have a resolution of $8 \times 8 \times 8$.

5.3 Modeling Surface Roughness

The actual particles in the powder bed are not perfectly spherical in shape. There are minor surface imperfections due to which the actual contact areas are not simply point contacts (in the case of touching spheres, used in the models in Section 5.2). Hence, the effects of surface roughness at the contact areas need to be considered for a realistic heat conduction model.

The geometric models for surface roughness are presented in Section 3.7. Using the same computational model as in Section 5.2, the ETC values

for the three packing arrangements were obtained. The last column in Table 5.1 shows the results from the surface roughness models. Notice that the values for the ETC are larger than the ones from idealized spherical models. This is due to the increase in the contact areas due to the surface roughness shapes (see Figure 3.3), which lead to more heat conduction paths.

5.4 Implementation Details

In the current implementation, an octree data structure is maintained internally to efficiently represent the geometrical details of the domain. This choice is primarily influenced by the significant speed-up that is achieved, in computational cost while computing the signed distance fields for complicated datasets. See Section 3.2 for details.

At each vertex of the volumetric representation, geometry information in the form of signed distance fields and computational information for the physical function that is modeled are stored. For each such parameter of the computational model, its values for both the current and next iteration (timestep) are stored.

In addition, if a vertex has a sign change edge, then additional information has to be stored to successfully impose the requisite boundary condition. However, in a typical dataset, only a few vertices need this extra storage for the BC. Hence, at each vertex, only a pointer to the data structure for the BC is stored, and memory is allocated only if needed. The aforementioned data structure for the BC stores the type of BC (Dirichlet, Neumann or Robin),

the value of the BC, the surface normal at the interpolated point H along the sign change edge and the interior point E as shown in Figure 4.1.

For visualization purposes, iso-surfaces from the volumetric representations using the standard marching cubes techniques are extracted, as outlined in Section 2.2.6. A hash table is maintained for the *sign change edges* in the boundary cells for fast retrieval. Since the volumes that are generated are *very* large scale, extracting the iso-surfaces from these volumes might be cumbersome and time consuming. The *contour spectrum* approach is used for narrowing down the range of interesting iso-values [10]. For complicated datasets, this library can be used to narrow the range of iso-values to choose from.

The final point of interest is the resolution of the volumes that are created as this depends on the features present in the input data. Finer features require higher Level-of-Detail (LoD) and hence, a larger resolution for the volume. Based on experience, a value of 64 as the octree resolution for most objects and 128 for fine features or thin shell elements are recommended [8, 9, 109].

The implementation is in C++ using a MSVC 6.0 compiler. The comprehensive geometric domain models for the surface roughness analysis (shown in Table 5.1) were developed on a PC equipped with a Pentium III 2.60 GHz processor with 512 MB of main memory and a GEForce FX 5200 graphics card. Interactive visualization was possible with this hardware for these datasets.

5.5 Time and Space Complexity Analysis

In this section, the time and space complexity for the multi-scale volumetric modeling framework are analyzed. A key parameter in this analysis is the grid resolution at each scale. While, this directly affects the spatial storage requirements for a given domain, this is also responsible for providing an upper bound for the timestep of the numerical solver.

First, the general values for arbitrary grid resolutions are computed and then some sample values are plugged in to estimate the spatial and computational requirements.

5.5.1 General Time and Space Complexity Requirements

Given a domain D, with a scale difference of 5, first an adaptive hierarchical recursive volumetric (geometric) representation of the domain needs to be created. Next for the computational simulations, the simulation parameters on this grid are defined and the computational requirements are estimated. Also, the total simulation period is assumed to be 100 seconds.

Let $\Delta x, \Delta y, \Delta z$ and Δt be the grid spacings in the X, Y and Z directions and the chosen timestep for the numerical simulation on this grid. For the sake of simplicity, the Domain D is assumed to be isotropic ($\Delta x = \Delta y = \Delta z$) and the same grid resolution g is chosen at each *level* in the recursive structure. Note that the assumption about the isotropy of the physical domain is valid in most situations.

For the stability of the numerical solver based on explicit FTCS schemes for solving the heat conduction problem in Equation 4.2, the relation given in [31], Equation 5.1 has to be satisfied.

$$(\frac{1}{\Delta x})^2 + (\frac{1}{\Delta y})^2 + (\frac{1}{\Delta z})^2 < \frac{1}{8} \times \frac{1}{\Delta t} \tag{5.1}$$

5.5.2 Spatial Storage

To achieve a multi-scale domain decomposition of 5 orders of magnitude, using recursive grid resolutions of g, the first calculation is to decide how many recursive sub-grid levels n the representation needs to have. For this Equation 5.2 is used.

$$(\frac{1}{g})^n = (\frac{1}{10})^5 \tag{5.2}$$

On simplification, this gives the number of sub-grids as in Equation 5.3

$$n = \log_g 10^5 \tag{5.3}$$

At each level of the recursive grid hierarchy, the number of voxels stored is g^3. Since there are n such grids, the total space requirement for the entire multi-scale volume grid is given in Equation 5.4.

$$\mathrm{O}_s(n) = (g^3)^n = (g)^{3n} \tag{5.4}$$

5.5.3 Time Complexity

From the constraint given in Equation 5.1, for an isotropic grid, $\Delta x = \Delta y = \Delta z = \frac{1}{g}$

$$g^2 + g^2 + g^2 < \frac{1}{8} \times \frac{1}{\Delta t} \tag{5.5}$$

On simplification, the timestep for the simulation for the given grid is given in Equation 5.6.

$$\Delta t < \frac{1}{3} \times \frac{1}{8} \times \frac{1}{g^2} < \frac{1}{24} \times \frac{1}{g^2} \tag{5.6}$$

Thus, the total number of time steps for one grid is $tsteps = 25 \times g^2$. At each time step, each of the voxels in the volume grid needs to be updated. Thus, the total number of computations is given in Equation 5.7. Note that this is actually a summation for each grid level, from the coarsest one to the the finest grid level.

$$\mathrm{O}_t(n) = \sum_{i=1}^{n} (g^3)^i \times (24 \times g^2)^i = \sum_{i=1}^{n} 24^i \times g^{5i} \tag{5.7}$$

Also, the expressions for time and space complexity given in Equations 5.7 and 5.4 are for the worst case scenarios. In practice, the hierarchical volume grids are adaptive, hence the run times and space storage requirements are less than predicted by these equations.

5.5.4 Analysis

The space complexity expression is exponential, and the time complexity is polynomial. In practice, the values of g need to be powers of 2, to build an octree[1]. Smaller values of g lead to very coarse and jaggy geometric approximations, while higher values need high processing times and more disk storage. Usually, values such as 32, 64, 128 and 256 are acceptable for most datasets [9, 109].

From the computational simulation perspective, at the interface where a grid cell is split into a sub-grid, the function values from the two cells need to be synchronized to satisfy the flux conservation law [16]. An abrupt difference in the grid resolution between adjacent cells leads to inaccuracies in the numerical solver. Hence, it is preferred to have smaller values for g like 2, 4 or 8. These values have been experimented with in [77], while they generally use only 2 or 4.

Thus, for the geometric modeling, values for g from 32 to 256 are generally possible, while, for the computational modeling, smaller values from 2 to 8 are needed. Thus, the total working range for considering the grid resolution g is from 2 to 256, in steps of powers of 2.

In practice, the spatial requirements as given by Equation 5.4 have to be multiplied with the actual amount of data stored at each voxel. The signed

[1]While, in general, other values of g are possible, they are usually re-sampled to the closest power of 2 for easier processing.

distance function value (for the geometric modeling) and the computational function values for the current and next time step, per physical parameter (for the computational modeling), are stored as floating points at each voxel. Assuming a 32 bit architecture and 4 bytes for a floating point number, the values from Equation 5.4 need to be multiplied with 3×4 to get the total space storage requirements in bytes.

With recent advances in computer hardware, computer memory is getting cheaper. Hence, the real bottleneck is the computational cost. Also, this varies exponentially with the grid resolution g. Hence, it is *faster* to have more levels of hierarchy and keep this number low. It is recommended to use values of grid resolution as high as possible at the coarsest grid, and for the subvolumes, to keep the grid resolution values between 4 and 8. This ensures fewer grid hierarchies as well as fewer subvolumes at each level.

5.6 Summary

The framework for multi-scale domain modeling based on volumetric representations has been applied to the SLS domain to model the heat transfer through the powder bed. The results are presented here and they are in close agreement with the available literature. The next chapter concludes the work.

Chapter 6

Closure

In this dissertation, research on creating both geometric and computational models for multi-scale domains has been presented. This framework was successfully applied to the SLS process.

6.1 Conclusions

A framework for creating comprehensive and realistic multi-scale geometric models is presented here. The input geometry descriptions are first converted into a common representation. Scale bridging is then performed to traverse the multiple length scales of the domain and then, using intuitive sculpting operations, the final geometric model is constructed.

The representation is then used in a physics driven simulation of the domain. A computational model is formulated and solved using numerical techniques. Thus, the resulting representations are validated with respect to the available literature to simulate simple physical processes. The models are in close agreement and also follow similar trends.

Several established techniques such as volumetric representations and volumetric sculpting have been leveraged to successfully establish a new paradigm

for modeling multi-scale domains. This work is a small step toward the eventual goal for realistically simulating such domains.

6.2 Future Work

This section outlines the various steps that are recommended to extend the current implementation to a more robust and all-purpose framework for creating multi-scale models and simulations.

6.2.1 Geometric Modeling

The current implementation can process only regular distance fields. This needs to be changed so that it can process adaptive distance fields [40, 80]. This would further reduce the memory (and also the computational) overheads associated with maintaining the function values at several grid points.

Note that the approach described above is for regular (uniform) volumetric grids only. However, in most cases, large regions of the regular grid are *empty* as they do not contain the embedded isosurface. To simply extract an isosurface, only the boundary (feature) cells are needed. Eliminating these vacant cells from the representation can save both the memory overhead in terms of storing them and also the computational expenditure to process them. Hence, an adaptive version of this scheme should be developed, where only the voxels that contain the regions of interest are stored. This is very similar to the work by Frisken et al. [40, 80].

6.2.2 Computational Modeling

A numerical solver based on the FTCS scheme (see Section 4.1 for details) has been used for the framework. Most computational multi-scale problems developed in practice are time dependant in nature. Hence, this work needs to be extended to include such domains. A fully implicit scheme instead of the current explicit scheme (the FTCS scheme as mentioned in Section 4.5) should be used to provide significant increase in efficiency and speedup, while still guaranteeing a measure of stability and convergence of the numerical scheme.

6.2.3 Volumetric Mesh Extraction

In some cases, it is advantageous to extract a high quality conforming volumetric mesh for the comprehensive domain [9]. For this, a scheme based on the work done by Zhang et al. [109] and Schneiders et al. [86, 87] is presented. While these schemes provide templates to create conforming hexahedral meshes from adaptive volumetric representations, in reality, the scheme gets very complicated when the neighboring coarse and fine grid cells differ by more than one level (for details, see Section 5.5 and Section 5.2 of [109]). Hence, when hexahedral meshes have to be generated, a maximum grid resolution value of 1 for all the subvolumes in the domain should be used. If the comprehensive domain representation is already created with a different grid resolution value, then the grid cells can be recursively collapsed to force this condition. Once this is done, then the templates provided by Scheiders [86, 87]

or Zhang et al. [109] can be used to extract the hexahedral meshes as desired.

Appendix

Appendix A

Modeling Boundary Conditions

This section presents a detailed description of the method to model the boundary conditions using volumetric grids.

A.1 Taylor Series

Consider the general case of a curved boundary intersecting the regular volumetric grid, shown in Figure A.1. First the simple 1D case is presented and then extrapolated to the general 3D case of interest. The reader is also advised to see [31] and [24] for further reading.

Using the Taylor series expansion at point B, the function at B is given by Equation A.1.

$$T(B) = T(P) + (\beta\Delta x)T_x(P) + \frac{(\beta\Delta x)^2}{2}T_{xx}(P) + O(\Delta x)^3 \tag{A.1}$$

Similarly, the function at point A is given by Equation A.2.

$$T(A) = T(P) - (\alpha\Delta x)T_x(P) + \frac{(\alpha\Delta x)^2}{2}T_{xx}(P) + O(\Delta x)^3 \tag{A.2}$$

Equations A.2 and A.1 form a set of simultaneous equations for $T_x(P)$ and $T_{xx}(P)$. These are expressed in matrix form in Equation A.3.

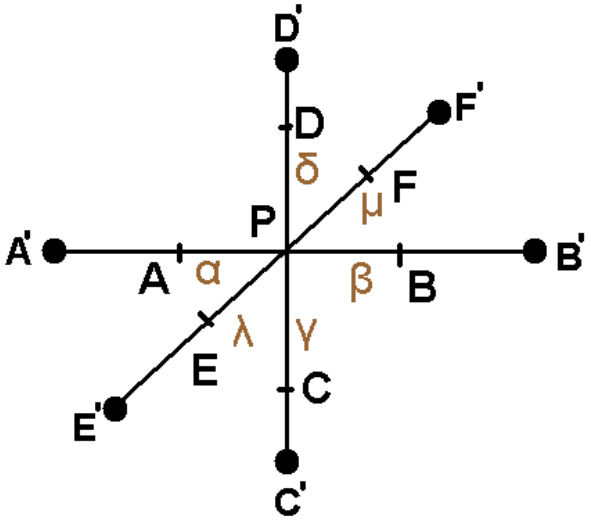

Figure A.1: Finite difference scheme at the boundary. Consider the 3D neighborhood of point P. The boundary points A, B, C, D, E and F are at distances $\alpha\Delta x$, $\beta\Delta x$, $\gamma\Delta y$, $\delta\Delta y$, $\lambda\Delta z$, $\mu\Delta z$, respectively from P.

$$\begin{vmatrix} T(B)-T(P) \\ T(A)-T(P) \end{vmatrix} = \begin{vmatrix} \beta\Delta x & \frac{(\beta\Delta x)^2}{2} \\ -\alpha\Delta x & \frac{(\alpha\Delta x)^2}{2} \end{vmatrix} \begin{vmatrix} T_x(P) \\ T_{xx}(P) \end{vmatrix} \tag{A.3}$$

Using Cramer's rule, the value of $T_{xx}(P)$ is given in Equation A.4.

$$T_{xx}(P) = \frac{\begin{vmatrix} \beta\Delta x & T(B)-T(P) \\ -\alpha\Delta x & T(A)-T(P) \end{vmatrix}}{\begin{vmatrix} \beta\Delta x & \frac{(\beta\Delta x)^2}{2} \\ -\alpha\Delta x & \frac{(\alpha\Delta x)^2}{2} \end{vmatrix}} \tag{A.4}$$

$$T_{xx}(P) = \frac{2\Delta x}{\alpha\beta\Delta x(\Delta x)^2} \frac{\begin{vmatrix} \beta & T(B)-T(P) \\ -\alpha & T(A)-T(P) \end{vmatrix}}{\begin{vmatrix} 1 & \beta \\ -1 & \alpha \end{vmatrix}} \tag{A.5}$$

$$T_{xx}(P) = \frac{2}{(\Delta x)^2}[\frac{\beta T(A) + \alpha T(B) - (\alpha+\beta)T(P)}{\alpha\beta(\alpha+\beta)}] \tag{A.6}$$

On simplification, this yields the expression for $T_{xx}(P)$ in Equation A.7.

$$T_{xx}(P) = \frac{2}{(\Delta x)^2}[\frac{T(A)}{\alpha(\alpha+\beta)} + \frac{T(B)}{\beta(\alpha+\beta)} - \frac{T(P)}{\alpha\beta}] \tag{A.7}$$

Note that this equation is identical to Equation 11.4.5 in [31] and Equations 7.8.5 and 7.8.6 (on page 498) in [24][1] . Also, substituting $\alpha = \beta = 1$ gives the well known form of the "3-point stencil" in 1D as Equation A.8. This is used in the frequent situation where the grid points are uniformly spaced.

$$T_{xx}(P) = \frac{1}{(\Delta x)^2}[T(A) + T(B) - 2T(P)] \tag{A.8}$$

Similarly, the finite difference equations in the Y and Z directions can also be written. These are given in A.9 and A.10.

$$T_{yy}(P) = \frac{2}{(\Delta y)^2}[\frac{T(C)}{\gamma(\gamma+\delta)} + \frac{T(D)}{\delta(\gamma+\delta)} - \frac{T(P)}{\gamma\delta}] \tag{A.9}$$

$$T_{zz}(P) = \frac{2}{(\Delta z)^2}[\frac{T(E)}{\lambda(\lambda+\mu)} + \frac{T(F)}{\mu(\lambda+\mu)} - \frac{T(P)}{\lambda\mu}] \tag{A.10}$$

[1]The authors show the derivation for the case of a curved boundary in 2D. The equation can be compared to Equation A.13 after ignoring all the terms arising from the Z dimension.

A.2 Steady State Heat Conduction Model

The steady state heat conduction (Laplace) equation is given in A.11. Now using A.7, A.9 and A.10 from above, this equation transforms into Equation A.12.

$$T_{xx} + T_{yy} + T_{zz} = 0 \tag{A.11}$$

$$\begin{aligned}\frac{2}{(\Delta x)^2}[\frac{T(A)}{\alpha(\alpha+\beta)} + \frac{T(B)}{\beta(\alpha+\beta)} - \frac{T(P)}{\alpha\beta}]+ \\ \frac{2}{(\Delta y)^2}[\frac{T(C)}{\gamma(\gamma+\delta)} + \frac{T(D)}{\delta(\gamma+\delta)} - \frac{T(P)}{\gamma\delta}]+ \\ \frac{2}{(\Delta z)^2}[\frac{T(E)}{\lambda(\lambda+\mu)} + \frac{U(F)}{\mu(\lambda+\mu)} - \frac{U(P)}{\lambda\mu}] = 0\end{aligned} \tag{A.12}$$

On simplification and re-arranging terms Equation A.13 is obtained. This is the steady state finite difference model. As mentioned before, the values of α, β, γ, δ, λ, μ are the fractions of the grid spacings in each of the directions, as shown in Figure 4.1.

$$\begin{aligned}[\frac{1}{\alpha\beta(\Delta x)^2} + \frac{1}{\gamma\delta(\Delta y)^2} + \frac{1}{\lambda\mu(\Delta z)^2}]T(P) = \\ \frac{1}{(\alpha+\beta)(\Delta x)^2}(\frac{T(A)}{\alpha} + \frac{T(B)}{\beta}) + \frac{1}{(\gamma+\delta)(\Delta y)^2}(\frac{T(C)}{\gamma} + \frac{T(D)}{\delta})+ \\ \frac{1}{(\lambda+\mu)(\Delta z)^2}(\frac{T(E)}{\lambda} + \frac{T(F)}{\mu})\end{aligned} \tag{A.13}$$

The values of α, β, γ, δ, λ, μ are equal to 1 at all the interior grid points. For such cases, the simplified version of Equation A.13 can be used, where the fractional values are all set to 1. This is given in Equation A.14.

$$[\frac{1}{(\Delta x)^2}+\frac{1}{(\Delta y)^2}+\frac{1}{(\Delta z)^2}]T(P) = \frac{T(A)+T(B)}{2(\Delta x)^2}+\frac{T(C)+T(D)}{2(\Delta y)^2}+\frac{T(E)+T(F)}{2(\Delta z)^2} \tag{A.14}$$

A.3 Transient State Heat Conduction Model

Consider the transient state heat conduction equation given in Equation A.15. Here, the symbol Ψ is used for thermal diffusivity.

$$T_{xx} + T_{yy} + T_{zz} = \frac{1}{\Psi}T_t \tag{A.15}$$

Now using A.7, A.9 and A.10 from above, this equation transforms into Equation A.16.

$$\begin{aligned}\frac{2}{(\Delta x)^2}[\frac{T^n(A)}{\alpha(\alpha+\beta)} + \frac{T^n(B)}{\beta(\alpha+\beta)} - \frac{T^n(P)}{\alpha\beta}] \\ + \frac{2}{(\Delta y)^2}[\frac{T^n(C)}{\gamma(\gamma+\delta)} + \frac{T^n(D)}{\delta(\gamma+\delta)} - \frac{U^n(P)}{\gamma\delta}] \\ + \frac{2}{(\Delta z)^2}[\frac{T^n(E)}{\lambda(\lambda+\mu)} + \frac{T^n(F)}{\mu(\lambda+\mu)} - \frac{T^n(P)}{\lambda\mu}] = \frac{1}{\Psi}\frac{U^{n+1}(P)-T^n(P)}{(\Delta t)}\end{aligned} \tag{A.16}$$

On simplification and re-arranging terms, Equation A.17 is obtained. This is the transient state finite difference model. As mentioned before, the

values of α,β,γ,δ,λ,μ are the fractions of the grid spacings in each of the directions, as shown in Figure A.1.

$$T^{n+1}(P) = \frac{2r_x}{(\alpha+\beta)}(\frac{T^n(A)}{\alpha} + \frac{T^n(B)}{\beta}) + \frac{2r_y}{(\gamma+\delta)}(\frac{T^n(C)}{\gamma} + \frac{T^n(D)}{\delta}) + \frac{2r_z}{(\lambda+\mu)}(\frac{T^n(E)}{\lambda} + \frac{T^n(F)}{\mu}) + [1 - \frac{2r_x}{\alpha\beta} - \frac{2r_y}{\gamma\delta} - \frac{2r_z}{\lambda\mu}]U^n(P) \quad \text{(A.17)}$$

where, $r_x = \frac{\Psi\Delta t}{(\Delta x)^2}, r_y = \frac{\Psi\Delta t}{(\Delta y)^2}, r_z = \frac{\Psi\Delta t}{(\Delta z)^2}$

Note that substituting $\alpha = \beta = \gamma = \delta = \lambda = \mu = 1$ gives the well known form of the *7-point stencil* in 3D in Equation A.18. This equation is used for the case of transient analysis of the 3D Heat equation where the grid points are uniformly spaced.

$$T^{n+1}(P) = r_x(T^n(A) + T^n(B)) + r_y(U^n(C) + T^n(D)) + r_z(T^n(E) + T^n(F)) + [1 - 2r_x - 2r_y - 2r_z]T^n(P) \quad \text{(A.18)}$$

Bibliography

[1] P. Adler and V. Mityushev. Effective Medium Approximation and Exact Formulae for Electrokinetic Phenomena in Porous Media. *Journal of Physics A: Mathematical and General*, 36(2):391–404, 2003.

[2] M. Aftosmis. Solution Adaptive Cartesian Grid methods for Aerodynamic Flows with Complex Geometries. *28th Computational Fluid Dynamics Lecture Series, Von Karman Institute for Fluid Dynamics, Waterloosesteenweg, Belgium*, 1997.

[3] Y. Akimov. Fields of Applications of Aerogels (Review). *Instruments and Experimental Techniques*, 46(3):287–299, 2003.

[4] A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome. A Conservative Adaptive Projection method for the Variable Density Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 142(1):1–46, 1998.

[5] A. Baerentzen. Octree based Volume Sculpting. *IEEE Visualization*, pages 9–12, 1998.

[6] M. Bahrami, J. Culham, M. Yovanovich, and G. Schneider. Review of Thermal Joint Resistance Models for Non-Conforming Rough Surfaces. *ASME Applied Mechanics Reviews*, 59(1):1–12, 2006.

[7] M. Bahrami, M. Yovanovich, and J. Culham. Effective Thermal Conductivity of Rough Spherical Packed Beds. *International Journal of Heat and Mass transfer*, 49(19-20):3691–3701, 2006.

[8] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. TexMol: Interactive Visual Exploration of Large Flexible Multi-Component Molecular Complexes. In *Proceedings of the IEEE Visualization 2004, Austin, Texas*, pages 243–250. IEEE Computer Society, 2004.

[9] C. Bajaj and L. Karlapalem. Volume Subdivision based Hexahedral Finite Element Meshing of Domains with interior 2-Manifold Boundaries. *ACM SIGGRAPH (AfriGraph), Cape Town, South Africa*, pages 127–136, 2006.

[10] C. Bajaj, V. Pascucci, and D. Schikore. The Contour Spectrum. In *Proceedings of the 1997 IEEE Visualization Conference, Phoenix, Arizona*, pages 167–173, 1997.

[11] C. Bajaj, J Warren, and G Xu. A Subdivision Scheme for Hexahedral Meshes. *The Visual Computer*, 18(5-6):343–356, 2002.

[12] C. Bajaj and Z. Yu. *Geometric Processing of Reconstructed 3D Maps of Molecular Complexes, in Handbook of Computational Molecular Biology, Edited by S. Aluru.* Chapman and Hall/CRC Press, Computer and Information Science Series, in press.

[13] I. Balberg and N. Binenbaum. Percolation Thresholds in Three dimensional Sticks System. *Physical Review Letters*, 52(17):1465–1468, 1987.

[14] R. Barzel. *Physically-Based Modeling for Computer Graphics.* Academic Press Inc., San Diego, California, 1992.

[15] J. Bear. *Dynamics of Fluid flow in Porus Media.* American Elseiver, New York, 1972.

[16] M. Berger. On Conservation at Grid Interfaces. *SIAM Journal of Numerical Analysis*, 24(5):967–984, 1987.

[17] M. Berger and P. Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics*, 82(1):62–84, 1989.

[18] M. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53:484–512, 1984.

[19] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.

[20] M. Bern, D. Eppstein, and J. Gilbert. Provably Good Mesh Generation. *31st IEEE Symposium on Foundations of Computer Science, St. Louis, Missouri*, 1:231–241, 1990.

[21] M. Berzins. Mesh Quality: A Function of Geometry, Error Estimates or Both? Technical report, 7th International Meshing Roundtable, Sandia National Laboratries, October 1998.

[22] J. Bloomenthal. *Introduction to Implicit Surfaces*. Morgan Kaufmann Inc., San Francisco, 1997.

[23] E. Butcher, E. Berg, and E. Kunkel. Systems Biology in Drug discovery. *Nature Biotechnology*, 22:1253–1259, 2004.

[24] B. Carnahan, H. Luther, and J. Wilkes. *Applied Numerical Methods*. John Wiley and Sons, New York, 1969.

[25] C. Chan and C. Tien. Conductance of Packed Spheres in Vacuum. *ASME Journal of Heat Transfer*, 95:302–308, 1973.

[26] V. Chandru, S. Manohar, and E. Prakash. Voxel based Modeling for Layered Manufacturing. *IEEE Computer Graphics and Applications*, 15(6):42–47, 1995.

[27] M. Chen and J. Tucker. Constructive Volume Geometry. *Computer Graphics Forum*, 19(4):281–293, 2000.

[28] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[29] S. Coquillart. Extended Free-Form Deformation: A Sculpting Tool for 3D Geometric Modeling. *ACM SIGGRAPH Computer Graphics*, 24(4):187–196, 1990.

[30] B. Cutler, J. Dorsey, L. McMillan, M. Muller, and R. Jagnow. A Procedural Approach to Authoring Solid Models. *ACM SIGGRAPH Computer Graphics*, 21(3):302–311, 2002.

[31] B. D'Acunto. *Computational Methods for PDE in Mechanics*. Advances in Mathematics for Applied Sciences - Vol. 67. World Scientific Publishing, Singapore, 2004.

[32] D. DeCarlo and D. Metaxas. Adaptive Shape Evolution using Blending. In *Proceedings of the International Conference on Computer Vision, Cambridge, Massachusetts*, pages 834–839, 1995.

[33] A. Duncan, G. Peterson, and L. Fletcher. Effective Thermal Conductivity within Packed beds of Spherical Particles. *ASME Journal of Heat Transfer*, 111:830–836, 1989.

[34] W. E and B. Engquist. Multiscale Modeling and Computation. *Notices of the AMS*, pages 1062–1070, 2003.

[35] T. Elvins. A survey of Algorithms for Volume Visualization. *ACM SIGGRAPH Computer Graphics*, 26(3):194–201, 1992.

[36] E. Ferley, M. Cani, and J. Gascuel. Practical Volumetric Sculpting. *The Visual Computer*, 16(7):469–480, 2000.

[37] J. Foley and A. Van Dam. *Fundamentals of Interactive Computer Graphics*. Addison-Wesley Publishing Company, 1990.

[38] G. Forsythe and W. Wasow. *Finite-Difference Methods for Partial Differential Equations*. John Wiley and Sons, New York, 1960.

[39] S. Fortune. Robustness Issues in Geometric Algorithms. *Applied Computational Geometry, Springer Lecture Notes in Computer Science*, pages 9–14, 1996.

[40] S. Frisken, R. Perry, A. Rockwood, and T. Jones. Adaptive Sampled Distance Fields: A general Representation of Shape for Computer Graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New Orleans, Lousiana*, pages 249–254. ACM Press, 2000.

[41] T. Galyean and J. Hughes. Sculpting: An Interactive Volumetric Modeling technique. *ACM SIGGRAPH Computer Graphics*, 25(4):267–274, 1991.

[42] L. Gao and Z. Li. Effective Medium Approximation for Two component Non-linear Composites with Shape Distribution. *Journal of Physics : Condensed Matter*, 15:4397–4409, 2003.

[43] E. Garboczi, K. Snyder, J. Douglas, and M. Thorpe. Geometric Percolation Threshold of Overlapping Solids. *Physical Review A*, 52(1):819–829, 1995.

[44] C.-C. Ho, Y.-H. Lu, H.-T. Lin, S.-H. Guan, S.-Y. Cho, R.-H. Liang, B.-Y. Chen, and M. Ouhyoung. Feature Refinement Strategy for Extended

Marching Cubes: Handling on Dynamic Nature of Real-time Sculpting Application. In *Proceedings of IEEE 2004 International Conference on Multimedia and Expo (ICME04), Taipei, Taiwan*, pages 855–858, 2004.

[45] C. Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann, Inc., San Francisco, 1989.

[46] W. Hsu, J. Hughes, and H. Kaufman. Direct Manipulation of Free-Form Deformations. *ACM SIGGRAPH Computer Graphics*, 26(2):177–184, 1992.

[47] P. Hui, X. Zhang, A. Markworth, and D. Stroud. Thermal Conductivity of Graded Composites: Numerical Simulations and an effective Medium Approximation. *Instruments and Experimental Techniques*, 46(3):287–299, 2003.

[48] ANSYS Inc. Canonsburg, PA, USA.

[49] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual Contouring of Hermite Data. *ACM SIGGRAPH Computer Graphics*, 21(3):339–346, 2002.

[50] L. Kobbelt. *Discrete Fairing : The Mathematics of Surfaces VII.* In T. Godman and R. Martin ed., Information Geometers Ltd., 1996.

[51] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature Sensitive Surface Extraction from Volume Data. In *SIGGRAPH '01: Pro-*

ceedings of the 28th annual conference on Computer graphics and interactive techniques, Los Angeles, California, pages 57–66. ACM Press, 2001.

[52] R. Krupiczka. Analysis of Thermal Conductivity in Granular Materials. *Journal of International Chemical Engineering*, 7(1):122–144, 1967.

[53] U. Labsik, K. Hormann, M. Meister, and G. Greiner. Hierarchical Iso-Surface Extraction. *Journal of Computing and Information Science in Engineering*, 2(4):323–329, 2002.

[54] S. Li. Adaptive Mesh Methods and Software for Time-Dependent Partial Differential Equations. *Doctoral dissertation, Department of Computer Science, University of Minnesota-Minneapolis*, 1996.

[55] W. Liu, H. Park, D. Qian, E. Karpov, H. Kadowaki, and G. Wagner. Bridging Scale Methods for Nanomechanics and Materials. *Computer Methods in Applied Mechanics and Engineering*, 195:1407–1421, 2006.

[56] C. Loop. Smooth Subdivision Surfaces based on Triangles. *Masters Thesis, Department of Mathematics, University of Utah*, August 1987.

[57] A. Lopes and K. Brodlie. Improving the Robustness and Accuracy of the Marching Cubes Algorithms for Isosurfacing. *IEEE Visualization and Computer Graphics*, 9(1):16–29, 2003.

[58] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[59] M. Mantyla. *An Introduction to Solid Modeling.* Computer Science Press, Maryland, 1998.

[60] E. Martensson. Modeling Electrical Properties of Composite Materials. *Doctoral dissertation, Department of Electrical Engineering, Division Electrotechnical Design, Stockholm*, 2003.

[61] E. Martensson and U. Gafvert. A Three dimensional Network Model describing a Non-linear Composite Material. *Journal of Physics D: Applied Physics*, 37:112–119, 2004.

[62] C. Maurer, R. Qi, and V. Raghavan. A Linear time Algorithm for computing exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions. *IEEE Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[63] D. J. Mavriplis. Multigrid approaches to Non-Linear Diffusion problems on Unstructured Meshes. Technical report, NASA/CR-2001-210660 ICASE, Report No. 2001-3, NASA Langley Research Center Hampton, VA, February 2001.

[64] K. McDonnell, Y-S. Chang, and H. Qin. DigitalSculpture: A Subdivision-based approach to Interactive Implicit Surface Modeling. In *Graphical*

Models, volume 67, pages 347–369, 2005.

[65] Z. Melek and J. Keyser. Multi-Representation Interaction for Physicaly Based Modeling. *Symposium on Solid and Physical Modeling, Cambridge, Massachusetts*, pages 187–196, 2005.

[66] S. Mitchell and S. Vavasis. Quality Mesh Generation in higher Dimensions. *SIAM Journal of Computing*, 29:1334–1370, 2000.

[67] K. Morton and D. Mayers. *Numerical Solution of Partial Differential Equations*. Cambridge University Press, New York, 1994.

[68] J. Munkres. *Topology: A First Course*. Englewood Cliffs, N.J., Prentice-Hall, New Jersey, 1974.

[69] Z. Neda, R. Florian, and Y. Brechet. Reconsideration of Continuum Percolation of Isotropically oriented Sticks in Three dimensions. *Physical Review E*, 59(3):3717–3720, 1999.

[70] H. Neeman. Autonomous Hierarchical Adaptive Mesh Refinement for Multiscale Simulations. *Doctoral dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign*, 1996.

[71] P. Ning and J. Bloomenthal. An Evaluation of Implicit Surface Tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993.

[72] J. Norrell. A Mixed Mode Thermal/Fluid Model for Imprrovements in SLS Part Quality, Machine Design and Process Design. *Doctoral*

dissertation, Department of Mechanical Engineering, The University of Texas at Austin, 2000.

[73] P. Oswald, W. Dahmen, and A. Kurdila. *Multiscale Wavelet Methods for Partial Differential Equations*. Academic Press, London, 1997.

[74] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi. Volumetric Illustration: designing 3D Models with internal textures. *SIGGRAPH, ACM Transactions in Graphics*, 23(3):322–328, 2004.

[75] S. Owen. A Survey of Unstructured Mesh Technology. *7th International Meshing Roundtable, Dearborn, Michigan*, pages 26–28, 1998.

[76] M. Parashar and J. Browne. Distributed Dynamic Data-Structures for Parallel Adaptive Mesh-Refinement. In *Proceedings of the International Conference on High Performance Computing, New Delhi, India*, 1995.

[77] M. Parashar and J. Browne. On Partitioning Dynamic Adaptive Grid Hierarchies. In *29th Annual Hawaii International Conference on System Sciences, Maui, Hawaii*, 1996.

[78] H. Park and W. Liu. An Introduction and Tutorial on Multiple Scale Analysis in Solids. *Computer Methods in Applied Mechanics and Engineering*, 193:1733–1772, 2004.

[79] B. Payne and A. Toga. Distance Field manipulation of Surface Models. *Computer Graphics and Applications, IEEE*, 12(1):65–71, 1992.

[80] R. Perry and S. Frisken. Kizamu: A System for Sculpting Digital Characters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques, Los Angeles, California*, pages 47–56. ACM Press New York, NY, USA, 2001.

[81] A. Rappoport, A. Sheffer, and M. Bercovier. Volume-preserving Free-Form Solid. In *Proceedings of the third ACM symposium on Solid modeling and applications, Salt Lake City, Utah*, pages 361–372. ACM Press, 1995.

[82] A. Raviv and G. Elber. Three Dimensional Freeform Sculpting via Zero sets of Scalar Trivariate Functions. *Computer Aided Design*, 32(8):513–526, 2000.

[83] A. G. Requicha and J. Rossignac. Solid Modeling and beyond. *IEEE Computer Graphics and Applications*, 12(5):31–44, 1992.

[84] R. Satherley and M. Jones. *Hybrid Distance Field Computation, In K. Mueller and A. Kaufman (eds.), Volume Graphics.* Springer-Wien, New York, 2001.

[85] R. Satherley and M. Jones. Vector-City Vector Distance Transform. *Computer Vision and Image Understanding*, 82(3):238–254, 2001.

[86] R. Schneiders. A Grid based Algorithm for the Generation of Hexahedral Mesh Elements. *Engineering with Computer*, 12:168–177, 1996.

[87] R. Schneiders, R. Schindler, and F. Weiler. Octree Based Generation of Hexahedral Mesh Elements. In *Proceedings of the 5th International Meshing Roundtable, Pittsburgh, Pennsylvania*, 1996.

[88] P. Schroder, D. Zorin, T. DeRose, L. Kobbelt, J. Stam, and J. Warren. Subdivision for Modeling and Animation. ACM SIGGRAPH 1999 Special Course Notes, 1999.

[89] T. Sederberg and S. Parry. Free-Form Deformation of Solid Geometric Models. *ACM SIGGRAPH Computer Graphics*, 20(4):151–160, 1996.

[90] R. Segura and F. Feito. Algorithms to test Ray-Triangle intersection, comparative study. In *Proceedings of the 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen (Czech Republic)*, 2001.

[91] James Albert Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press, Cambridge, U.K., 2^{nd} edition, 1999.

[92] J. Shewchuk. What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Proceedings, 11th International Meshing Roundtable, Ithaca, New York*, pages 115–126, 2002.

[93] C. Sigg, R. Peikert, and M. Gross. Signed Distance Transform using Graphics Hardware. In *Proceedings of the 14th IEEE Visualization Conference, Seattle Washington*, 2003.

[94] V. Singh, D. Silver, and N. Cornea. Real-Time Volume Manipulation. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics, Tokyo, Japan*, pages 45–51, 2003.

[95] V. Spitzer. The Visible Human Male: A Technical Report. *Journal of Americal Medical Informatics Association*, 3(2):118–130, 1996.

[96] M. Sramek and A. Kaufman. Alias-free Voxelization of Geometric Objects. *IEEE Visualization and Computer Graphics*, 5(3):251–267, 1999.

[97] J. Stiles and T. Bartol. *Monte Carlo Modeling of Realistic Cellular Structure and Physiology.* Computational Neuroscience: Realistic Modeling for Experimentalists, Editor E. De. DeSchutter, CRC Press, 2001.

[98] M. Sun and J. Beaman. A Three Dimensional Model for Selective Laser Sintering. In *Proceedings of the 1991 Solid Freeform Fabrication Symposium, Austin, Texas*, pages 62–69, 1991.

[99] K. Tai. Simulations on many Scales: The Synapse as an example. *Pure Applied Chemistry*, 76(2):295–302, 2004.

[100] P. Tallec. Domain Decomposition methods in Computational Mechanics. In *Computational Mechanics Advances*, volume 1(2), pages 121–220, 1994.

[101] S. Teng and S. Wong. Unstructured Mesh Generation: Theory, Practice and Perspectives. *International Journal of Computational Geometry and Applications*, 10(3):227–266, 2000.

[102] Y.-H. Tsai. Rapid and Accurate Computation of the Distance Function using Grids. *Journal of Computational Physics*, pages 175–195, 2002.

[103] S. Wang and A. Kaufman. Volume Sculpting. In *Proceedings of the 1995 symposium on Interactive 3D graphics, Monterey, California*, pages 151–156. ACM Press, 1995.

[104] G. Xu, Q. Pan, and C. Bajaj. Discrete Surface Modeling using Geometric Flows. Technical report, Department of Computer Sciences, The University of Texas at Austin, August 2003.

[105] S. Xue and J. Barlow. Models for the Prediction of Thermal Conductivity of Powders. In *Proceedings of the 1991 Solid Freeform Fabrication Symposium, Austin, Texas*, pages 62–69, 1991.

[106] S. Yagi and D. Kunii. Studies of Effective Thermal Conductivities in Packed Beds. *Journal of American Institute of Chemical Engineers (AICHE)*, 3(3):373–381, 1957.

[107] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with Poisson-based Gradient Field manipulation. *SIGGRAPH, ACM Transactions in Graphics*, 23(3):644–651, 2004.

[108] L. Zhang, W. Liu, S. Li, D. Qian, and S. Hao. Survey of Multi-Scale Meshfree Particle Methods. *Lecture Notes in Computational Science and Engineering (LNCSE)*, 26:441–458, 2002.

[109] Y. Zhang and C. Bajaj. Adaptive and Quality Quadrilateral/Hexahedral Meshing from Volumetric Data. *Special issue of Computer Methods in Applied Mechanics and Engineering (CMAME)*, 195(9-12):942–960, 2006.

Vita

Lalit Chandra Sekhar Karlapalem was born in Bombay (now Mumbai), India in October 1977. Until the turn of the century, he was in Pune, India finishing up his schooling and gathering two years of work experience in industry.

He entered graduate school at The University of Texas at Austin in Fall 2000. After finishing his Master's degree in Fall 2002, he continued as a doctoral student in Mechanical Engineering at the same institution.

Permanent address: 7117 Woodhollow drive, Apt. 1512
Austin, Texas 78731

This dissertation was typeset by the author.