Analysis of Functionally Graded Material Object Representation Methods

by

Todd Robert Jackson

B.S.E., Princeton University (1994)
S.M., Massachusetts Institute of Technology (1997)

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Author .....
Department of Ocean Engineering
January 28, 2000

Certified by .....
Nicholas M. Patrikalakis, Ph.D.
Kawasaki Professor of Engineering
Thesis Supervisor

Certified by .....
Emanuel M. Sachs, Ph.D.
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by ...
Nicholas M. Patrikalakis
Chairman, Departmental Committee on Graduate Studies

Analysis of Functionally Graded Material Object Representation Methods

by

Todd Robert Jackson

Submitted to the Department of Ocean Engineering
on January 28, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Solid Freeform Fabrication (SFF) processes have demonstrated the ability to produce parts with locally controlled composition. To exploit this potential, methods to represent and exchange parts with varying local composition need to be proposed and evaluated. In modeling such parts efficiently, any such method should provide a concise and accurate description of all of the relevant information about the part with minimal cost in terms of storage. To address these issues, several approaches to modeling Functionally Graded Material (FGM) objects are evaluated based on their memory requirements.

Through this research, an information pathway for processing FGM objects based on image processing is proposed. This pathway establishes a clear separation between design of FGM objects, their processing, and their fabrication. Similar to how an image is represented by a continuous vector valued function of the intensity of the primary colors over a two-dimensional space, an FGM object is represented by a vector valued function spanning a Material Space, defined over the three-dimensional Build Space. Therefore, the Model Space for FGM objects consists of a Build Space and a Material Space. The task of modeling and designing an FGM object, therefore, is simply to accurately represent the function $\mathbf{m}(\mathbf{x})$ where $\mathbf{x} \in$ Build Space.

Data structures for representing FGM objects are then described and analyzed, including a voxel-based structure, finite element method, and the extension of the Radial-Edge and Cell-Tuple-Graph data structures with FGMDomains in order to represent spatially varying properties. All of the methods are capable of defining the function $\mathbf{m}(\mathbf{x})$ but each does so in a different way. Along with introducing each data structure, the storage cost for each is derived in terms of the number of instances of each of its fundamental classes required to represent an object.

In order to determine the optimal data structure to model FGM objects, the storage cost associated with each data structure for representing several hypothetical models is calculated. Although these models are simple in nature, their curved geometries and regions of both piece-wise constant and non-linearly graded compositions reflect the features expected to be found in real applications. In each case, the generalized cellular methods are found to be optimal, accurately representing the intended design.

Thesis Supervisor: Nicholas M. Patrikalakis, Ph.D.
Title: Kawasaki Professor of Engineering

Thesis Supervisor: Emanuel M. Sachs, Ph.D.
Title: Professor of Mechanical Engineering

Dedication

This dissertation is dedicated to my wife, Courtney, and our families. Thank you for your unwaivering love and support as I pursued my dream.

Acknowledgements

I wish to thank my thesis supervisors, Professors **Nicholas M. Patrikalakis** and **Emanuel M. Sachs**, for the opportunity to participate in this research project. Participating in a project forging new grounds in Computer Aided Design and Manufacturing has been an exciting, challenging, and rewarding process. I am grateful for their mentoring and patience as we explored new ideas together and defined the direction of my research. **Professor Michael J. Cima** also deserves recognition for his role in helping to define the direction of this work. I would also like to thank and recognize the efforts of my colleagues who contributed invaluable energy and insights to the challenges presented through this work, including **David Brancazio**, **Dr. Wonjoon Cho**, **Hongye Liu**, and **Dr. Hauijun Wu**. Along with my thesis supervisors, Dr. Cho's critical readings of the drafts of this manuscript have been instrumental in expediting the completion of this document and ensuring that the final draft meets the expectations for an MIT dissertation.

The author and investigators in this project also gratefully recognize the financial support of the **National Science Foundation (grant #DMI9617750)** and the **Office of Naval Research (grant #N00014-96-1-000857)**, without which this research would not have been possible.

The examples used in this thesis were generated through the CAD system *SolidWorks* and meshed using the Finite Element Analysis package *Algor*.

Finally, I wish to thank my friends and colleagues for their friendship and support through my years at MIT. The names are too numerous to mention here, but their comradery deserves just as much acknowledgement. Please forgive me for not attempting to list names here, for to do so would inevitably lead to the omission of someone who should not have been excluded. I choose rather to close these acknowledgements with a simple "thank you" and trust they all will realize how much their friendships mean to me and take some satisfaction in seeing this work in final form.

Thank you.

Contents

List of Tables

List of Figures

Symbols

$\lceil a \rceil$ The ceiling of a; the first integer b equal to or larger than a such that $0 \leq b - a < 1$

$\lfloor a \rfloor$ The floor of a; the first integer b equal to or less than a such that $0 \leq a - b < 1$

A_p the area of the surface p.

G^0 position continuity: no breaks or gaps exist over a G^0 curve or surface

G^1 tangent continuity: the tangent vector for a G^1 curve varies continuously; the orientation of the tangent plane for a G^1 surface varies continuously

G^2 curvature continuity: the center of curvature for a G^2 curve varies continuously; the principal curvatures for a G^2 surface vary continously

M^0 material continuity: the volume fraction of each material varies continuously over an M^0 region,

$$\lim_{\Delta \mathbf{x} \to \mathbf{0}} (\mathbf{m}(\mathbf{x}) - \mathbf{m}(\mathbf{x} + \Delta \mathbf{x})) = \mathbf{0}$$

M^1 material derivative continuity: the rate of change of the volume fraction of each material varies continuously within an M^1 region, $\vec{\nabla}\mathbf{m}(\mathbf{x})$ is continuous at all points $\mathbf{x}$ in an M^1 region

$\mathbf{M}$ a vector or array or composition points

S_a storage cost for an object of type a

S_{bln} storage cost for a Boolean object

S_{flt} storage cost for a float object

S_{int} storage cost for an integer object

S_{ms} storage cost for a Material System

S_{ptr} storage cost for a pointer object

R radius of curvature

$\mathbf{U}$ a vector or arracy of points in parameter space

V_q the volume of the region of space q

$\mathbf{X}$ a vector or array of points in Build Space

$\mathbf{X}_{n\times m}$ multidimensional array of data (mn floats or doubles)

b Boolean flag (true or false)

d_m number of materials in the Material System; dimension of the Material Space in which the FGM object exists

$\mathbf{m}$ a composition stored in an FGM modeling system; a vector of volume fractions of materials in Material System; the modeled point in Material Space

$\mathbf{m}^*$ the intended composition to be stored in an FGM modeling system; the intended point in Material Space

n_λ number of intensity levels represented in a voxel-based data structure (each intensity level maps to a specific volume fraction of material)

$\mathbf{u}$ a point in parameter space

$\mathbf{x}$ a geometric point in Build Space

ϵ_g accuracy in approximating a desired or ideal geometry

ϵ_m accuracy in approximating desired or ideal composition

κ_g curvature of curve or surface

κ_m material curvature

$\vec{\nabla} f(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0}$ gradient of $f(\mathbf{x})$ at $\mathbf{x}_0$;

$$\vec{\nabla} f(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_0} = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)\Big|_{\mathbf{x}=\mathbf{x}_0}$$

$\vec{\nabla} f(\mathbf{x}) \cdot \hat{\mathbf{v}}\Big|_{\mathbf{x}_0}$ directional derivative of $f(\mathbf{x})$ at point $\mathbf{x}_0$ in direction $\hat{\mathbf{v}}$;

$$\left(\vec{\nabla} f(\mathbf{x}) \cdot \hat{\mathbf{v}}\right)\Big|_{\mathbf{x}=\mathbf{x}_0} = \left(\frac{\partial f(\mathbf{x})}{\partial \hat{\mathbf{v}}}\right)\Big|_{\mathbf{x}=\mathbf{x}_0}$$

Chapter 1

Introduction

1.1 Motivation

With recent advances in Solid Freeform Fabrication (SFF), the ability to fabricate parts with Local Composition Control (LCC) is becoming a reality, opening the door to creating a whole new class of parts with graded compositions. Despite the advanced capabilities of these SFF machines, access to this new technology is limited by how information is represented, exchanged, and processed. Designers need new CAD representations to capture their ideas as models with graded compositions and manufacturers need algorithms capable of converting these models into machine instructions for their fabrication. A method for maintaining this information, however, has not yet been adopted as the preferred solution from the many approaches to representing volumetric data. This presents an obstacle to the exploration of tools for capturing design intent, algorithms for processing models for fabrication, and finally exercising the capabilities of LCC to produce FGM parts and tooling, as each method maintains data differently and follows a different paradigm. One of the major obstacles to choosing a solid modeling method through which to explore modeling FGM objects is the memory required to accurately store information within the model. By investigating the memory requirements for various approaches to representing parts with graded compositions, a decision about which method should be preferred as the basis for the solid modeling of FGM parts can be made.

1.2 Scope of research

1.2.1 Thesis

With recent advances in Solid Freeform Fabrication (SFF) technology to achieve Local Composition Control (LCC), Computer Aided Design (CAD) methods need to extended to truly realize the potential of Functionally Graded Material (FGM) parts and tools. To better understand the CAD

issues involved with modeling FGM objects, this dissertation identifies and compares several likely candidate data structures in terms of how they might represent FGM objects. Through this work, the following hypothesis is examined:

> "A memory efficient and accurate approach to modeling FGM objects can be achieved by extending solid modeling methods currently used for mechanical design. This extension consists of incorporating methods to map from a Build Space to a Material Space into their underlying generalized cellular decomposition or B-rep data structures."

Through the exploration of this hypothesis, the information flow from concept to fabrication is outlined, modeling issues for FGM objects are identified, several alternative approaches to FGM modeling are discussed, and sample FGM objects are presented and their expected storage costs are analyzed in terms of each data structure.

1.2.2 Approach

In order to study the memory required to model FGM objects, this dissertation will begin with an overview of what FGM objects are and how they can be fabricated through SFF processes. Next, a review of methods for model exchange is presented followed by proposed solutions for FGM modeling. The decision for selecting one method over another remains an open question at this point. To answer it, issues concerning the modeling of FGM objects are outlined, including the representation of geometry and composition and how this information should be processed into machine instructions for fabrication through SFF. The accuracy of representation of both the geometry and composition are identified as the relevant parameters in evaluating modeling methods and a clean separation is established between the representation of the object and its processing. To address the issue of representation, data structures for maintaining FGM models are then introduced (voxel-based, triangulated shells, finite element based, and generalized cellular decomposition), along with their memory requirements in terms of the data they maintain. Since several of the modeling methods approximate the designer's intent, a relationship is established between the number of instances of each of their data classes with the accuracy at which the intended geometry and composition is captured. Since working with FGM objects is a new concept, a set of design tools are proposed for defining FGM models as pathways for capturing the designer's intent for material variation. In order to explore the storage costs associated with modeling FGM objects, several hypothetical FGM objects are introduced, using these tools, containing features that a real FGM part might possess (such as curved surfaces and nonlinearly graded compositions). The storage costs associated with representing the objects in each modeling method are analyzed, as functions of accuracy in shape and composition representation. This dissertation then concludes with a recommendation for a preferred solid modeling method based on memory considerations and suggests possible directions

for future research.

1.3 Solid Freeform Fabrication

Solid Freeform Fabrication (SFF) refers to a class of manufacturing processes that build objects in an additive fashion directly from a computer model. While some SFF processes are restricted to building in a single material at a time, most can be adapted to exercise some degree of control over the local composition [17, 35, 57]. Such Local Composition Control (LCC) provides the opportunity to design and create parts with graded composition tailored for specific applications. These graded compositions have become known as Functionally Graded Material (FGM) [20, 80] and refer to the tailored distribution of material within a part designed for optimal performance.

The capability of LCC and fabrication of FGM parts could be utilized by a wide variety of industries. Applications could range from multi-color visualization models to functional parts and tools. For example, the potential to convey additional tissue information through multi-color medical models would increase their usefulness in medical applications such as surgical planning [31]. Not limited to prototyping, SFF processes can also build finished parts and tools. The possibility of incorporating graded compositions could be used in applications requiring the optimization of the mechanical properties of parts and tools at a local scale, potentially reducing distortion due to internal stresses, increasing hardness at points of greatest wear, or resisting failure, for example by locally controlled toughening [80]. The application of FGM compositions to drug delivery devices is even being studied as a means to achieve optimal, controlled release of medicine into a patient [78].

1.3.1 Single material SFF processing

SFF processes build parts by repeatedly adding minute primitives of material according to a computer model until the final object is created [3, 39]. Although there are many variations on this process using different materials and mechanisms for adding material, the underlying philosophy is the same: fabricate an object in an additive fashion directly from a computer model. Some of these SFF processes, each capable of building parts in some additive fashion such as Selective Laser Sintering (SLS) [6], Laminated Object Manufacturing (LOM) [10], Stereolithography (SLA) [34], Shape Deposition Manufacturing [74], Selective Area Laser Deposition (SALD) [35], and 3D Printing(3DP) [57]. To illustrate how SFF processes work, the 3D Printing process is explained here.

3D Printing manufactures a part by selectively binding powder together according to a computer model. The build cycle begins by spreading a layer of powder over the print bed. A print head then traverses the bed, selectively depositing binder [1] over the regions corresponding to the interior

[1] Any liquid that can be delivered through a printhead (similar to an inkjet printhead) that binds powder together.

of a slice of the computer model. After the layer is printed, the print bed is lowered and another layer of powder is spread. The process of spreading powder, depositing binder, and lowering the print bed is repeated, as shown in Figure 1-1(a), until the entire volume of the object is printed. At the end of the process, the bound powder becomes the manufactured object, effectively rendering the computer model as a physical object. Post processing steps may also be performed, included sintering and infiltration, in order to increase the density and strength of the object. Currently, metal and ceramic parts are being manufactured through 3D Printing, but the potential exists to build with any material supplied in powder form. A collection of parts manufactured with a single material is pictured in Figure 1-1(b).

1.3.2 Local Composition Control through SFF with multiple materials

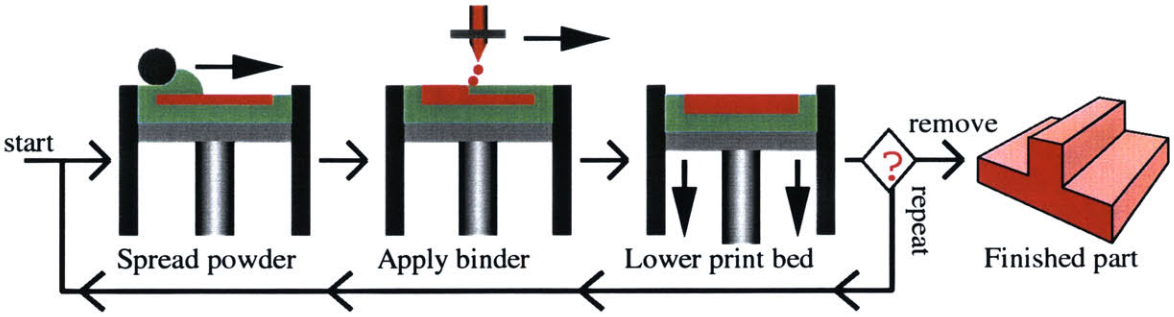
Considering that most SFF processes build parts on a point-wise basis, a analogy can be made between how parts are constructed and how images and documents are rendered by the computer on display or hard-copy devices. Following this analogy, the capability to generate mixtures of materials similar to how varying intensities of color are generated through image processing is a logical next step for SFF processes. Through the local control of primitives of different material, the material in a part can be controlled on a local scale, achieving what has become known as Local Composition Control (LCC). Although many SFF processes are capable of achieving LCC, how 3D Printing could achieve it is explained here.

Similar to how an ink-jet printer prints color documents, 3D Printing can achieve LCC with multiple materials. This is accomplished by using a print-head with several jets, as shown in Figure 1-2, each depositing binders and/or slurries [2] of unique material. By varying the pattern in which the jets deposit material on the powder-bed, the material composition can be controlled on the scale of the binder droplets ($\approx 100\mu m$). Regions of uniform and graded composition can be created in a manner analogous to how continuous tone images are rendered on a hard-copy device from primary colors. With this capability, graded compositions can be designed along with the geometry of the part, tailoring the part's physical properties for a specific purpose or function.

1.3.3 Modeling and processing for SFF

Solid Freeform Fabrication (SFF) processes build parts directly from computer models. These models may originate from sampled volumetric data or solid models of parts designed within a CAD system. The processing of these models for fabrication is unique to each SFF system (depending on the architecture and mechanism of adding material) but the general philosophy can be understood by looking at the information flow for 3D Printing, as shown in Figure 1-3.

[2] A solution or colloid without binding properties, used as a transport mechanism for locally adding material to an FGM object.



(a)

(b)

Figure 1-1: (a) 3D Printing illustrating how SFF processes can build a part on a point-wise basis. (b) Parts fabrication through 3D Printing demonstrating the flexibility of SFF to produce complex geometries. [59]

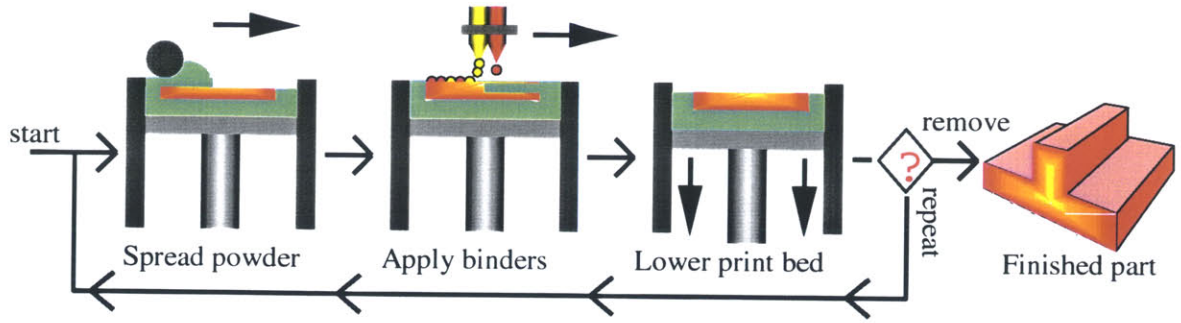


Figure 1-2: 3D Printing illustrating Local Composition Control by selectively depositing droplets of different material into a powderbed to form a Functionally Graded Material object.

The process begins with a designer interacting with a CAD system to define the shape of the object. The designer uses tools to modify the geometry of the part and the CAD system provides visual and numerical feedback to the designer about the status of the design. The solid model is represented internally by a proprietary CAD representation, which may be exchanged with other designers and manufacturers using a similar system or translated into a neutral file format (such as STEP [49, 29] or IGES [27]) for exchange between different systems.

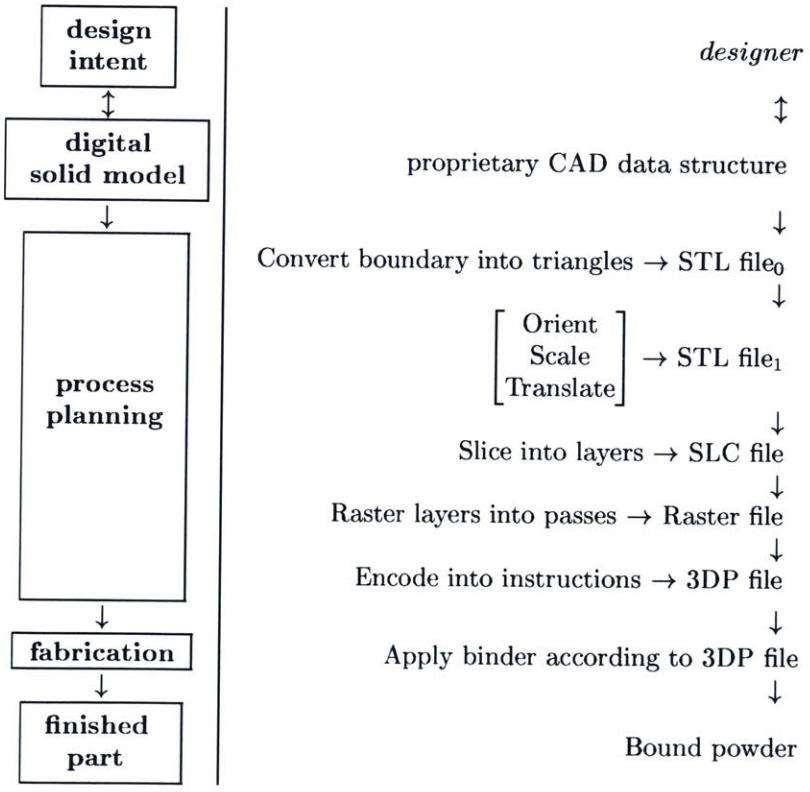


Figure 1-3: Information flow for 3D Printing.

Once the shape of the part is completely defined, process planning begins, generating the in-

structions for the fabrication process. The first step in this process is the conversion of the model from its native representation into a tessellation of triangles approximating its boundary. Next, this tessellated model is oriented, scaled, and positioned in order to ensure its optimal fabrication. Once correctly positioned, the model is sliced into layers of polygons. These polygons are then rasterized into scan lines which are then encoded into instructions for the 3D Printer, indicating where to apply binder as the printhead traverses the printbed.

Although capable of achieving Local Composition Control, the pathway for information flow in current practice does not permit either the definition of FGM objects at the design stage or the conveyance of the material variation downstream to the printer.

Chapter 2

Previous work

2.1 Introduction

Computer Aided Design and Manufacturing methods aim to improve the design process and the quality of the finished process by using computers to represent, design, and evaluate models and from these models, generate plans for their accurate fabrication. Initially, CAD systems were simply used to reproduce traditional drafting tools within the computer, enabling the creation and storage of digital engineering drawings. As computers and modeling methods evolved, CAD systems evolved as well, from simply representing engineering drawings to allowing the complete definition and representation of solid parts and assemblies in three dimensions, along with all the relevant manufacturing data (surface finish, bill of materials, etc.). Relative to Solid Freeform Fabrication (SFF), the capability of solid modeling has enabled designers to create models and directly manufacture the parts through an SFF process, in a manner analogous to producing hardcopy output directly from a word processing program. One of the next challenges for solid modeling is the representation of graded compositions, or Functionally Graded Material (FGM). With their representation, FGM objects could then be designed and fabricated through SFF processes capable of Local Composition Control (LCC). The current standards for data exchange do not allow this, as explained below in the review of current data exchange methods, providing motivation for research into new modeling methods. To address this shortcoming, several solutions for modeling FGM parts have been proposed by different groups and are also outlined below.

2.2 Current data exchange methods

Data exchange allows designers to communicate models between each other, the customer, and manufacturers. Although there are as many methods to define an exchange specification as there

are CAD systems, two paths are considered here. The first, the STL file format, is introduced as the accepted specification for model processing within the SFF community. The second, STEP (ISO10303), represents a neutral exchange standard proposed by the CAD community at large to enable the complete and exact exchange of model information between various CAD systems.

2.2.1 STL file

In current practice within the SFF community, the processing of models for fabrication is based on the STL file format. This format was introduced in 1987 by the Albert Consulting Group and remains the most common specification for the transfer of models to SFF processing systems [42, 47]. Designers use the tools within a commercial solid modeling system to define a model. Once a model is defined, the boundary of the model is tessellated into a collection of triangular facets, approximating the intended design of the part, and written to a file in the STL format. This file is then used as input to the data processor for generating machine instructions for the part's fabrication. At this point, the model may be oriented, scaled, or support structures added to ensure optimal fabrication. The tessellated model is then sliced into layers and instructions unique to each SFF machine for printing each layer are written to a file. This process of information flow is illustrated in Figure 1-3.

To represent a part, the STL format maintains a triangulated representation of the model's boundary. The file is organized as a list of solids, each of which maintains a list of facets defining its boundary, as shown in Figure 2-1. The orientation of the facet is defined by: (1) the ordering of the vertices such that they form a counter clockwise circuit when viewed from outside the solid and (2) the outward pointing normal.

The appeal of the STL file to the SFF community comes from its simplicity. Since the STL file is simply a list of triangles, algorithms to read, write, manipulate, and process models are relatively simple to develop and implement (when compared to generalized boundary representation methods). With the convenience of simplicity, however, comes penalties as well. These penalties can be divided into three general categories: approximation tradeoffs, shortcomings in format definition, and conversion errors [47].

Through the representation of solid models as a triangulated boundary representation within the STL format, curves and curved surfaces are approximated, resulting in a loss of information and accuracy. To provide a higher fidelity representation of a designer's intent for curved objects, more triangles must used, resulting in greater storage cost. Therefore, with the use of a triangulated boundary representation, a tradeoff must be made in the accuracy of representation of the model and the memory required to represent that model.

The definition of the STL format, itself, introduces problems. First, the STL file is unnecessarily verbose, containing redundant data. The orientation of each facet, for instance, is doubly defined by the ordering of its vertices and the associated normal. Second, truncation errors are introduced into

STL File Format
solid <name>
facet normal $\hat{n}_x\ \hat{n}_y\ \hat{n}_z$
outer loop
vertex $x_1\ y_1\ z_1$
vertex $x_2\ y_2\ z_2$
vertex $x_3\ y_3\ z_3$
endloop
endfacet
facet normal $\hat{n}_x\ \hat{n}_y\ \hat{n}_z$
outer loop
vertex $x_1\ y_1\ z_1$
vertex $x_2\ y_2\ z_2$
vertex $x_3\ y_3\ z_3$
endloop
endfacet
$\vdots$
endsolid <name>

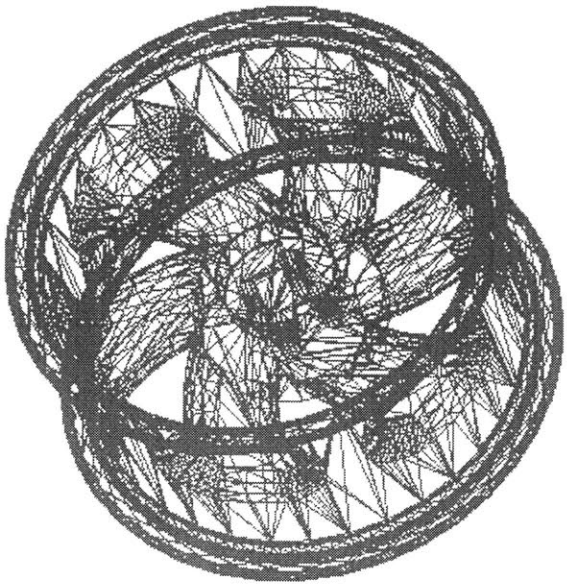
Figure 2-1: STL file format. The boundary of a model is described as a list of triangular facets and their normals.

the representation through the use of single precision floating point numbers. This may result in gaps between facets that cannot be easily detected due to the third problem with the format: lack of topological information. Although the topology, or connectivity, between facets may be inferred, the lack of explicit topology prevents models from being quickly validated as being solids; *ie.* having watertight boundaries. STL models containing gaps and holes may be created and not detected without the costly step of re-deriving all of the neighbor information between triangles [4, 5].

The final categories of problems with the STL format are due to conversion from the original solid model into the triangulated representation. During triangulation from the original model into the STL file, any of the following errors may be generated: gaps (open shells) or dangling facets formed, degenerate facets recorded, inconsistent normals assigned, or the topology of the model altered [41, 47, 11]. All of these errors are a function of the accuracy and robustness of the algorithms used to convert the native solid modeling representation with the designer's CAD system into the STL file.

To illustrate the STL format, Figure 2-2(a) shows the tessellated representation of a wheel (with a radius of 269 mm), initially designed with swept, freeform surfaces. Figure 2-2(b) is a truncated description of the part as an STL file. The STL model of the wheel has a total of 26906 triangular facets with a reasonable approximation tolerance, requires 3.17 Megabytes to store the data in ASCII format (1.34 Megabytes as a binary file). The model was created and converted to the STL specification using the commercial CAD system SolidWorks ^{TM} [1].

[1] Solidworks: http://www.solidworks.com



```
solid Wheel
  facet normal -1.000000e+000 0.000000e+000 0.000000e+000
    outer loop
      vertex 7.095000e+001 2.913194e+002 7.026579e+001
      vertex 7.095000e+001 2.914028e+002 7.636772e+001
      vertex 7.095000e+001 3.106206e+002 8.149973e+001
    endloop
  endfacet
  facet normal -1.000000e+000 0.000000e+000 0.000000e+000
    outer loop
      vertex 7.095000e+001 3.106206e+002 8.149973e+001
      vertex 7.095000e+001 2.914028e+002 7.636772e+001
      vertex 7.095000e+001 2.882984e+002 1.048139e+002
    endloop
  endfacet
  facet normal -1.000000e+000 0.000000e+000 0.000000e+000
    outer loop
      vertex 7.095000e+001 3.106206e+002 8.149973e+001
      vertex 7.095000e+001 2.882984e+002 1.048139e+002
      vertex 7.095000e+001 2.795565e+002 1.320610e+002
    endloop
  endfacet
  facet normal -1.000000e+000 0.000000e+000 0.000000e+000
    outer loop
      vertex 7.095000e+001 2.685262e+002 2.101446e+002
      vertex 7.095000e+001 2.845330e+002 1.940968e+002
      vertex 7.095000e+001 2.647845e+002 1.974923e+002
    endloop
                .
                .
                .
  facet normal -1.000000e+000 0.000000e+000 0.000000e+000
    outer loop
      vertex 7.095000e+001 2.647845e+002 1.974923e+002
      vertex 7.095000e+001 2.845330e+002 1.940968e+002
      vertex 7.095000e+001 3.011244e+002 1.720122e+002
    endloop
  endfacet
endsolid
```

(a) (b)

Figure 2-2: (a) Faceted approximation of a wheel (26906 triangles). (b) STL text file describing part (3.17 Megabytes in ASCII format, 1.34 Megabytes in binary format)

2.2.2 STEP: STandard for the Exchange of Product model data

The concept of enterprise data management has become increasingly important in every facet of the business world, with the goal of promoting the complete and neutral data exchange across an organization. The STEP standard was developed for representing product data for data exchange throughout a product's design, analysis, and manufacture cycles [28]. Initiated in 1984, STEP was intended to become an international standard based on existing specifications established by various countries, such as IGES (U.S.), PDDI (U.S.), SET (France), NEDO (U.K.), and VDA/VDMA-FS (Germany) [49]. To improve upon these specifications, the following goals for STEP were established:

Completeness: the capability to maintain all relevant product data over its life-cycle, from inception to retirement.

Extensibility: able to be expanded to incorporate future representations, processes, products, and methodologies.

Testability of additions: clear procedures for validating extensions to the standard.

Efficiency: permit product description and storage with minimal cost.

Compatibility: provide as much compatibility as possible with existing standards.

Minimal redundancy: allow only one way of representation for each concept.

Computing environment independence: enable the exchange of data across systems and architectures.

Logical classification of data elements: organize the standard such that future expansions can easily build on established data elements.

Implementation validation: establish a set of tests and protocols for validating new implementations using STEP, ensuring their correctness.

Today, STEP represents a comprehensive set of international standards for modeling product life cycles across a broad spectrum of industries. While a great deal of the standard has been established, it continues to grow as the new requirements for emerging products and processes are identified, building upon existing parts of the standard.

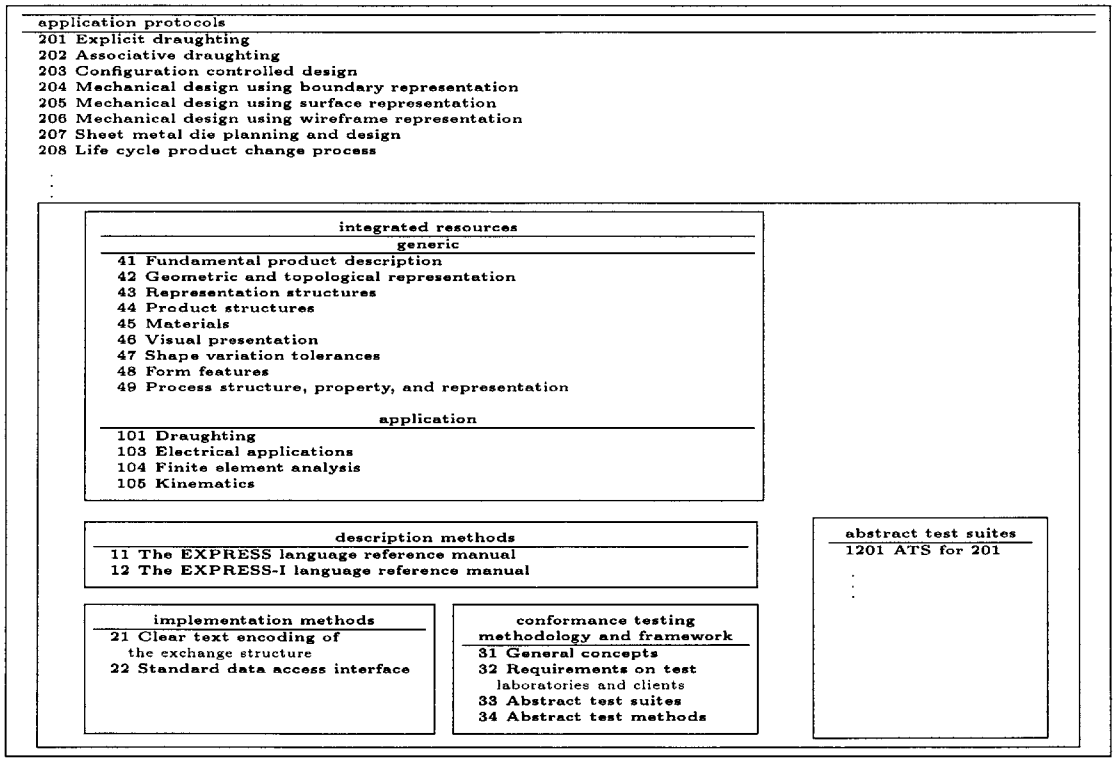


Figure 2-3: The structure of STEP.

STEP is comprised of six major classes, as shown in Figure 2-3 [29]. The top most class, *application protocols*, define a protocol for the exchange of data for a specific application. These protocols are developed using parts of the *integrated resources.* The integrated resources, in turn, are described in terms of *description methods*, for which the schema language EXPRESS is used [60, 75]. In order to ensure compliance with the standard, a STEP implementation must satisfy the specifications of

the three remaining classes: implementation methods, conformance testing methodology, and the abstract test suite.

The basis of solid modeling within STEP is Part 42 of the integrated resources: geometric and topological resources [30]. STEP separates the implementation of the geometry resources from the topology. Points, curves, and surfaces are each individually defined and then positioned and oriented using the transformation operators. A wide variety of parametric entities for geometric representation are included in the standard, permitting accurate description of a model. The hierarchy of these entities is illustrated in Figure 2-4(a). The connectivity between these geometric entities is maintained through an associated topological graph, consisting of the classes shown in Figure 2-4(b). An object recorded as a STEP file is shown in Figure 2-5. Figure 2-5 shows the surfaces of a boundary representation of the wheel blade. Figure 2-5(b) is the description of the part as a STEP file, containing a precise mathematical desciption of all of the curves and surfaces as well as the connectivity between all of the topological entities. In this representation, the model consists only 269 faces and requires 11.1 Megabytes of memory to store the data in text format. Although larger than the corresponding text representation of the wheel in STL format (see Figure 2-2(a), the STEP description provides a complete and accurate description of wheel's design, including analytic descriptions of the features (curves and surfaces) using trimmed NURBS patches and how they are related to each other. This detailed information in a neutral format allows the exchange of the product data to another system without loss of information. The model was an example provided with the commercial CAD system SolidWorks ^{TM}, and translated into the STEP standard through SolidWorks TM export interface.

2.3 Modeling of FGM objects

The modeling of parts with graded material compositions requires the attachment of composition information to the model. Although there are existing methods for composite structures [73, 72], the capability to model graded composition requires that a representation go a step beyond decomposing models into subregions of uniform material. The following describes several of the approaches proposed in the literature for modeling these FGM objects.

2.3.1 Voxel-based modeling

The use of voxel-based or exhaustive enumeration methods are commonly used for representing volumetric data sampled from the real-world [37]. The structure of a voxelized model consists of cells uniformly distributed over a 3D grid. Each cell maintains information about its composition. Two classes of voxel-based methods exist: binary voxel and grey-value [37]. In the former class, a binary value is assigned to each voxel, indicating whether it corresponds to a cell interior or exterior

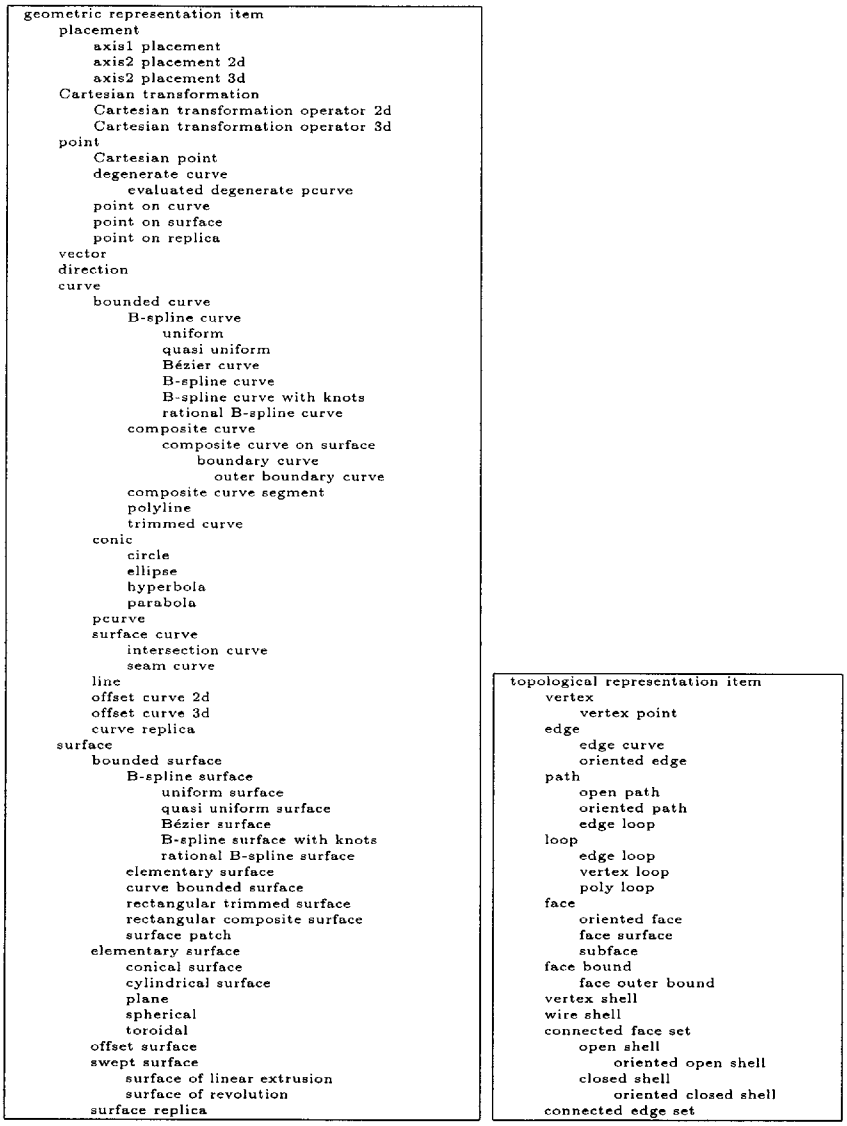


Figure 2-4: (a) Geometric and (b) topological entities defined with Part 42 of STEP (ISO10303).

(a)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION (( 'STEP AP203' ),
        '1' );
FILE_NAME ('wheel.step',
        '1999-11-11T17:16:43',
        ( 'Jackson' ),
        ( 'MIT Design Lab' ),
        'SwStep 1.0',
        'SolidWorks 98011',
        '' );
FILE_SCHEMA (( 'CONFIG_CONTROL_DESIGN' ));
ENDSEC;

DATA;
#1= EDGE_CURVE ( 'NONE', #70750, #70945, #32869, .T. ) ;
#2= ORIENTED_EDGE ( 'NONE', *, *, #56432, .T. ) ;
#3= ORIENTED_EDGE ( 'NONE', *, *, #75214, .F. ) ;
#4= CARTESIAN_POINT ( 'NONE', ( -0.02277223799738261100, ... ) ;
#5= CARTESIAN_POINT ( 'NONE', ( -0.02274808706025059600, ... ) ;
#6= CARTESIAN_POINT ( 'NONE', ( -0.02276491948137489600, ... ) ;
#7= CARTESIAN_POINT ( 'NONE', ( -0.02278175190249919600, ... ) ;
#8= CARTESIAN_POINT ( 'NONE', ( -0.02283956758760015800, ... ) ;
#9= CARTESIAN_POINT ( 'NONE', ( -0.02293799242417949100, ... ) ;
#10= CARTESIAN_POINT ( 'NONE', ( -0.02303641726075882400, ... ) ;
#11= CARTESIAN_POINT ( 'NONE', ( -0.02317545069753161600, ... ) ;
#12= CARTESIAN_POINT ( 'NONE', ( -0.02335419307990956000, ... ) ;
#13= CARTESIAN_POINT ( 'NONE', ( -0.04164771119154294500, ... ) ;
#14= CARTESIAN_POINT ( 'NONE', ( -0.04092825589242170300, ... ) ;
#15= CARTESIAN_POINT ( 'NONE', ( -0.04020880261889587400, ... ) ;
                    .
                    .
                    .
```

(b)

Figure 2-5: (a) Surfaces defining the boundary of a wheel (269 surfaces). (b) STEP encoding of the part (11.1 Megabytes).

to the modeled object. Grey-value go one step further, in which a value is assigned to each voxel representing the density of a material or property assigned to that cell. Sources of real world data for voxel models come from sampling methods that naturally generate data over a grid, such a X-ray computed tomography, magnetic resonance, positron emission tomography, computed radiography, digitized x-rays, and ultrasound [62, 77].

A summary of the advantages and disadvantages of voxel-based modeling methods has been provided by Kaufman *et al* [38], which is reproduced here:

- Disadvantages of voxel-based models:

  Discrete form Voxel-based methods provide finite resolution, approximating surfaces and volumes as discrete primitives. This artifact of voxel-based methods also complicates transformations and results in information loss during the transformations.

  Loss of geometric information: Since a model is represented as discrete information, information about specific surfaces and features is not readily available for rendering and design algorithms.

  Memory and processing: In order to represent models accurate, large sets of voxels are required.

- Advantages of voxel based models:

  Insensitivity to scene complexity: All objects in a scene are represented as collections

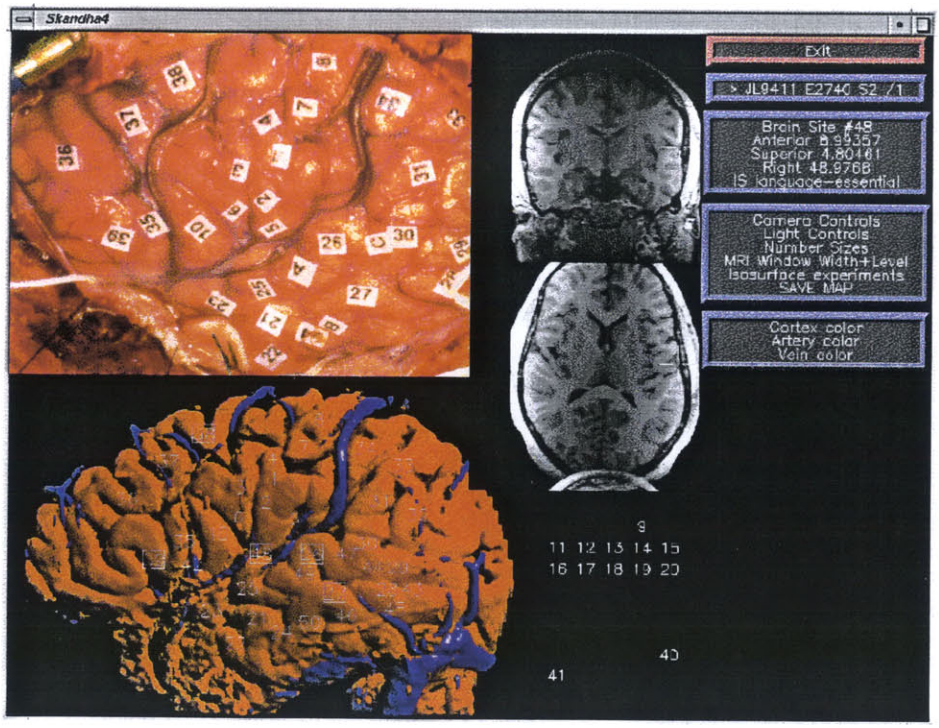


Figure 2-6: Photograph of actual brain, magnetic resonance imaging of brain, voxelized model of brain. (image courtesy of University of Washington Structural Informatics Group http://sig.biostr.washington.edu/) [77]

> voxels in an ordered grid, allowing direct rendering without concern for intersections between polygons (as in surface models).

Citing the fact that raster based graphics have dominated in 2D environments over vector based approaches [19], Kaufman suggests that a similar migration is possible in the 3D modeling as memory prices drop and processing speed increases [38].

Citing the close resemblance between how voxel-based methods represent parts and how SFF processes build parts, several researchers have suggested the use of voxel-based modelers for SFF model representation and design [9, 46]. Chandru *et al* introduces the Geometric Workbench for Rapid Prototyping (G-WoRP), incorporating a voxel-based data structure for modeling parts with design tools for capturing designers' intent [9]. In their approach, they suggest that the process planning for SFF fabrication is greatly simplified through the use of a voxelized representation. The voxels in their data structure correspond directly with the material primitives added during the build of the part, reducing the intermediate steps of generating an STL file, slicing it, and rasterizing the slices. They also suggest the possibility of building composite structures by associating material information with each voxel, thereby enabling the representation and design of parts with local composition.

Despite advocating the use of voxelized models for modeling parts for SFF, Chandru *et al* [9]

recognize that "the memory requirements of voxel models are enormous" [9]. They provide as an example that a $400 \times 400 \times 400$ grid requires 6.4×10^7 bytes of memory. To address this issue, they suggest trading computation time with memory cost by compression, octrees, shells, or maintain "the original geometric representation and use voxelization algorithms when necessary" [9].

2.3.2 Finite element modeling

In Computer Aided Engineering (CAE), designs and physical problems are often analyzed through the Finite Element Method [2], allowing the study and prediction of engineering properties throughout a finite element model. With its ability to represent volumetric data, the finite element modeling is another avenue for establishing a representation for FGM objects.

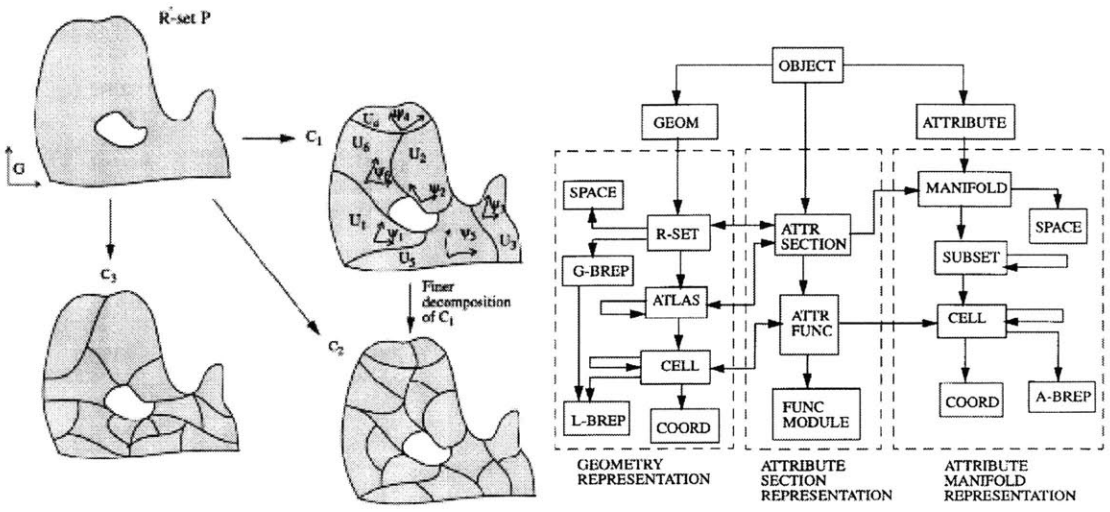
In order to perform a finite element analysis, a simplified model of the problem is formed, consisting of many finite elements (tetrahedra, bricks, etc.) interpolating nodes in space. These nodes are positioned such that the shape of the mesh of elements captures the important features of the problem being studied. Boundary and load conditions are then imposed on the appropriate nodes and a set of governing equations are solved over the elements, determining the values at all of the nodes, solving for the properties of interest throughout the model.

With its ability to represent volumetric data, the Finite Element Method has been proposed as a solution for the representation and design of FGM objects, most notably by Pegna [50] at a presentation in the 1998 Solid Freeform Fabrication Symposium. Pegna suggested that models represented by point sets used in existing finite elements analysis systems could be used to model spatially varying material distributions within FGM objects. Although proposed as a method to represent FGM objects, Pegna does not give a predication of the memory requirements for modeling FGM objects in such a manner.

2.3.3 Generalized modeling methods

While the preceding approaches to FGM modeling adopt analysis methods for volumetric data to the design of FGM objects, another avenue to solving this problem is to extend solid modeling methods currently used in Computer Aided Design (CAD) systems to handle graded material data.

The following sections introduce two possible approaches. The first proposes the extension of Boundary Representation (B-rep) modeling to include material information. The second proposes a generalized cellular decomposition scheme. Upon close review of the two methods, it will become apparent that both approaches advocate the decomposition of models into general regions over which material variations are mapped.



(a) (b)

Figure 2-7: (a) Proposed decomposition of solid model into *atlases*. (b) Proposed data structure for r_m-object modeling based [40].

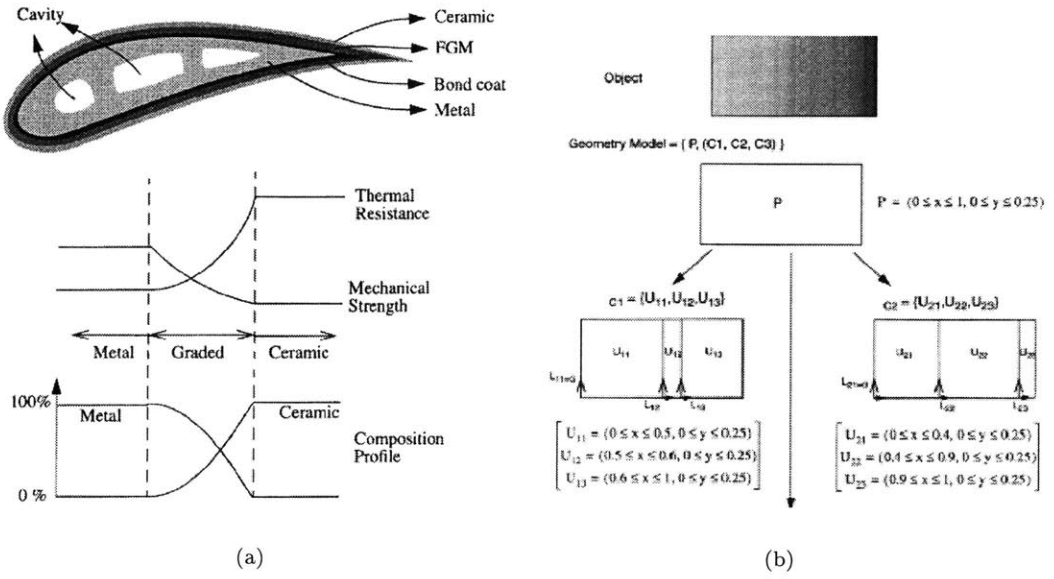
r_m-objects

Kumar *et al.* [42, 43, 40] have proposed the extension of solid modeling to handle heterogeneous objects by using ($r-$sets) with information about material variation (r_m-sets). The shape of a model is defined in terms of an $r-$set (a form of B-rep modeling). This *geometry model* is further decomposed into *atlases* over which material variations are mapped, as shown in Figure 2-7(b). In such a manner, FGM solid models are represented as r_m-objects, maintaining the geometry of the model's boundary and the volume fraction variation of each material throughout the object. In their initial proposal (1997), Kumar and Dutta [42], r_m-objects were to be represented in a data structure built upon the commercially available solid modeling kernel ACIS [65]. In their latest published work (1999), a new cellular data structure (Kumar *et al* [40]) is proposed to maintain solid models decomposed into cells with atlases defining their compositions and properties (see Figure 2-7(b)).

In order to define FGM models, Kumar *et al.* also propose a set of Boolean operators for modifying the geometry and composition stored in the data structure. Examples of FGM objects modeled as r_m-objects are given in Figures 2-8(a) and (b). They do not, however, provide an analysis of the memory required to store r_m-objects.

FGM cellular decomposition

A generalized cellular decomposition approach to FGM solid modeling was proposed by Jackson *et al.* [33, 32, 58]. In their approach, a solid model is decomposed into a collection of cells representing



(a) (b)

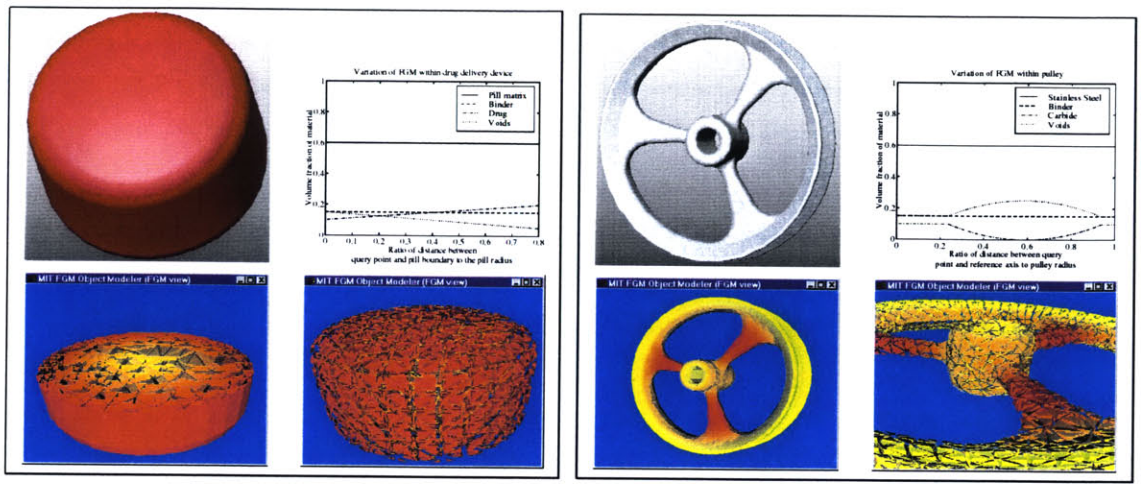
Figure 2-8: (a) Model motivating need for representation method for heterogeneous objects with r_m-sets [40]. (b) Example of a graded bar represented as an r_m-object, decomposed into cells [40].

the vertices, edges, faces, and regions of the solid model. The topology (or connectivity between the cells) is maintained through the use of a data structure known as the Cell-Tuple data structure [7, 8, 25, 26]. The geometry of each cell was maintained separately, similar to how the geometry is maintained in existing B-rep modelers. Along with the geometry of each cell, material information was also stored, allowing the modeling of both curved geometries and graded compositions for each cell.

In their prototype implementation, the shape and composition of the cells were defined in terms of Bézier curves, triangles and tetrahedra (linear in shape, but of any degree in composition). Control points blended according to the barycentric Bernstein polynomials define the shape of each cell. Likewise, control compositions defined in material space and another set of Bernstein basis functions define the material variation of each cell.

Although Jackson *et al* [32, 33] state that their approach is general in nature, the examples they provide were generated through a commercial meshing system [2] as an initial approach to subdividing models into sub-regions (see Figure 2-9). Acknowledging the expense of storing meshes in a relational database maintaining all of the topological data explicitly, Jackson *et al.* suggest that further research should address methods to more logically subdivide models as well as the development of additional cell classes for representing shape and composition.

[2] Algor: http://www.algor.com



(a) (b)

Figure 2-9: (a) A pulley consisting of graded material defined as a cellular model with Bézier triangles and tetrahedra. (b) A drug delivery device defined as a cellular model with Bézier triangles and tetrahedra. [33, 32].

2.4 Discussion

This chapter has provided a brief overview of the state-of-the-art in solid modeling relevant to modeling FGM objects.

For data exchange, the STL file format is recognized as the *de facto* specification in the SFF community. Serving the same purpose (data exchange) but with more accuracy and wider acceptance, the STEP standard was presented as the state-of-the-art in solid model exchange. Given the limitations of the STL format, the logical choice for data exchange for solid models should be STEP (provided that both the transmitter and receiver of the model posses the facilities to handle this format). For representing FGM objects, however, neither STL or STEP allow the definition of regions of graded compositions, motivating the exploration of extensions to STEP to handle graded materials.

Three classes of solid modeling for handing FGM objects were then introduced. Each addresses the issue of material variation in space differently. The first two, voxel-based and finite element based are commonly used for engineering analysis of volumetric data, but their use in mechanical design is not as common. The generalized methods suggest extending the concepts behind existing solid modeling methods used for design. The most common method, B-rep modelers, describe the boundary of a solid in terms of a shell of faces, defined based on a wide variety of surface types. Both Kumar *et al* and Jackson *et al* suggest that this paradigm can be followed for FGM object modeling in which models are decomposed into regions that map material variations, similar to how

shape is represented. By mapping each region into a material space, their compositions are defined.

Although several methods for representing FGM models have been introduced, no conclusions have been drawn on the storage costs associated the representing an FGM object in each.

Chapter 3

Identification of issues in FGM modeling

3.1 Motivation

Before proposing a solution for modeling FGM objects, the issues involved in the design, representation, and processing of FGM models must be understood. Only the most fundamental issues concerning the representation of FGM objects are presented here: the definition of shape and composition in terms of a Build Space and Material Space. These concepts are then put in the context of the overall information flow for SFF processing with LCC, clearly a separation between FGM modeling and SFF fabrication.

3.2 Modeling shape versus shape and composition

3.2.1 Geometric modeling

Modeling the geometry (or shape) of an object is the primary purpose of most design systems. The goal is simple: provide a system capable of maintaining a model's geometry and provide the tools to define that geometry by accurately capturing the designer's geometric intent. Existing CAD systems provide this functionality in many formats, but the basic idea is typically to define the boundary of an object, whether it is a single part and a component of an assembly.

The Model Space maintained in the data structure can be considered the Build Space in R^3 in which the model will be fabricated (see Figure 3-1(a)). The boundary of the model must clearly divide this Build Space into the interior and exterior of the model. To accomplish this, any of a number of solid modeling methods may be used (voxel-based, CSG, B-rep, cellular decomposition,

etc).

The designer modifies the boundary of the object through a set of design tools (extrude, cut, sweep, revolve, etc.) to generate the desired shape of the model. In addition to representing the object, information about the geometric nature of the boundary may be requested. Geometric interrogations may consist of calculating quantities such as surface normals, curvature, minimum distance between features, surface area, or volume. Although material information may be associated with a geometric model, it is added after the fact as an attribute to the enclosed region and is assumed to be uniform throughout that region (see Figure 3-1(b)).

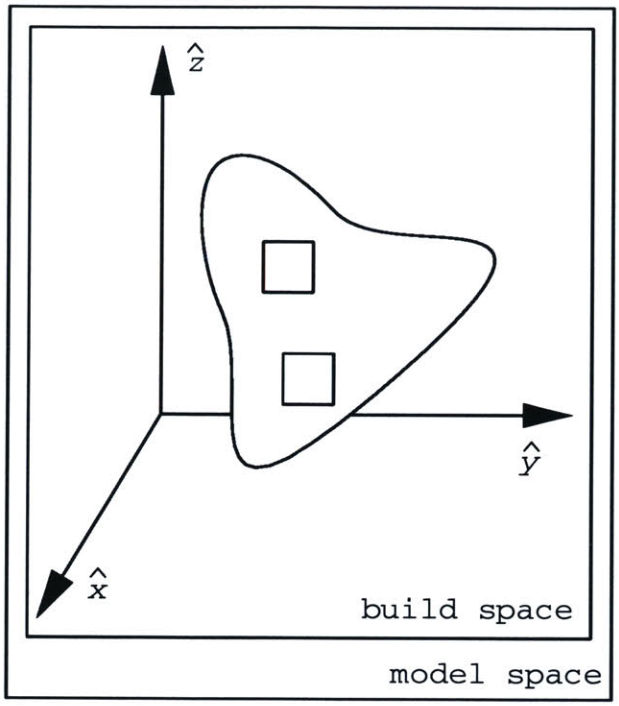
3.2.2 Geometric and material modeling

With the goal of modeling graded compositions, the issue of material modeling becomes as important in the representation of an object as the geometric modeling of its boundary. The Model Space in which the model is defined can now be considered as the combination of the Build Space and the Material Space (see Figure 3-2).

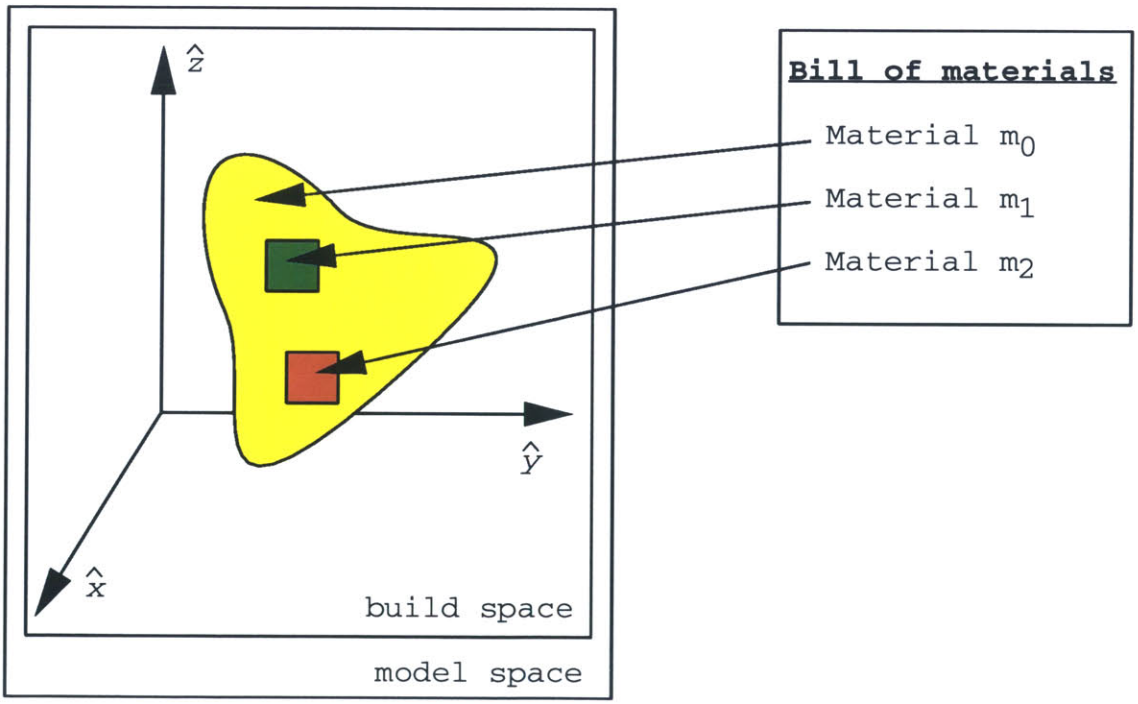
Again, the Build Space is simply the three dimensional space in which the object is to be fabricated. The Material Space, however, is spanned by the primary materials in the material system. This concept is analogous to the blending of primary colors (Cyan, Yellow, Magenta, and Black) in an ink-jet printer to produce a wide range of colors and tones for color hard-copy output. To achieve Local Composition Control, an SFF process builds a part by selectively adding varying quantities of different base materials. These materials comprising the material system create a Material Space out of which an FGM is defined. The dimension of the Material Space (d_m) is the number of materials out of which the object is to be composed.

To define an FGM object, a mapping from the Build Space ($\mathbf{X}$) into the Material Space ($\mathbf{M}$) must be provided. This concept has previously been suggested by Kumar *et al.* [42, 43] (through the use of *atlases*) and Jackson *et al.* [33, 32] (through the use of parametric cells with *control points* in Build Space and control compositions in Material Space). This concept of mapping from Build Space into Material Space is illustrated in Figure 3-3.

Before even exploring how one might maintain a digital representation of the model, it must be understood that the underlying goal of any FGM modeling method is to define a function spanning the material space for all the points in Build Space. This is accomplished by defining the concept of a composition, represented by $\mathbf{m}$, which defines the volume fraction of each of the primary materials



(a)



(b)

Figure 3-1: (a) The Model Space represented by state-of-the-art solid modeling systems. (b) Associating of materials to regions within a model.

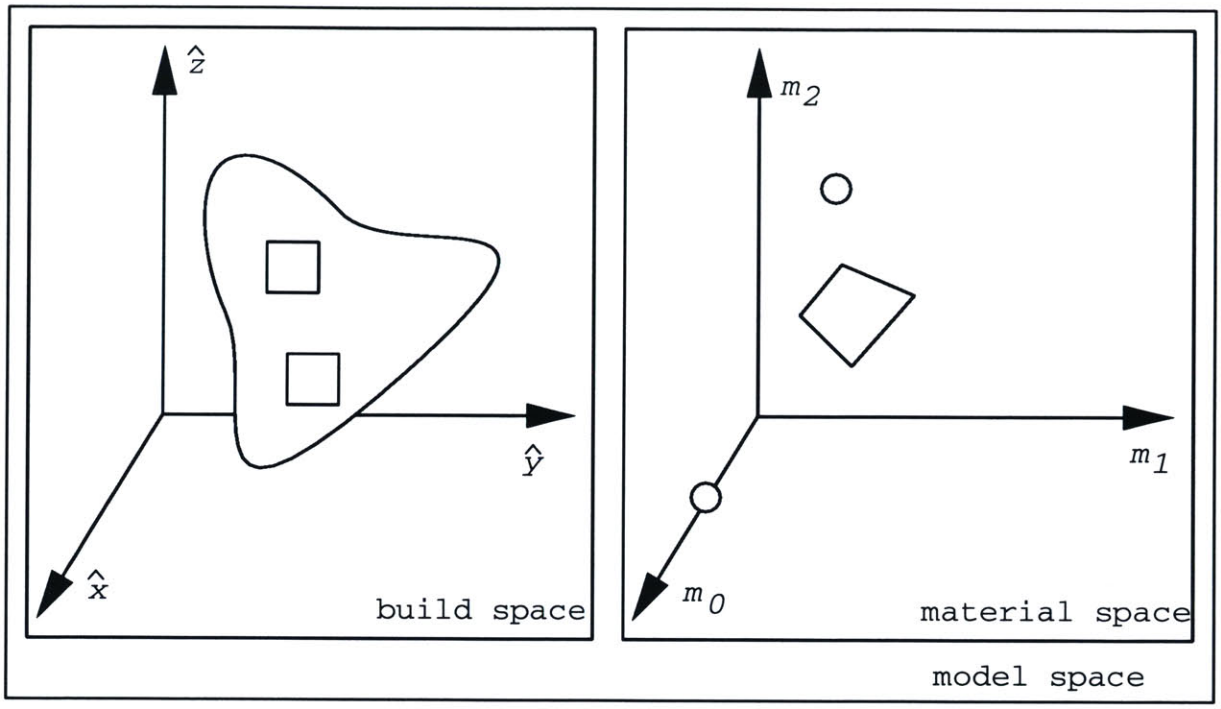


Figure 3-2: The Model Space for objects consisting of graded material spans both the Build Space ($\mathbf{X}$) and a Material Space ($\mathbf{M}$) in which the material variations are defined.

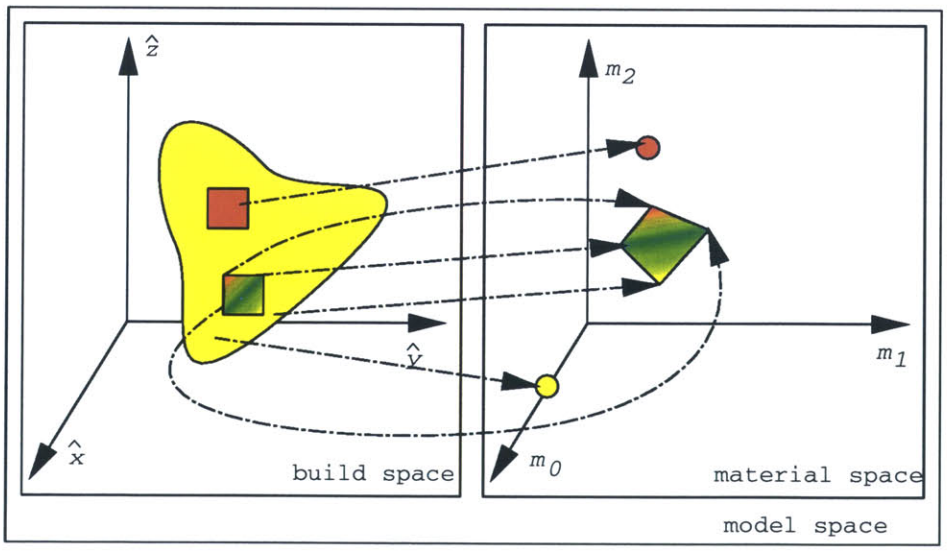


Figure 3-3: To define an FGM object, each point in the Build Space ($\mathbf{x} \in \mathbf{X}$) must map to a composition in the Material Space ($\mathbf{m}(\mathbf{x}) \in \mathbf{M}$)

at every point in the Build Space.

$$\mathbf{M} = \text{material}_0 \times \text{material}_1 \times \text{material}_2 \ldots \tag{3.1}$$

$$\mathbf{m}(\mathbf{x}) = \begin{bmatrix} \%\text{material}_0 \\ \%\text{material}_1 \\ \%\text{material}_2 \\ \vdots \end{bmatrix} \in \mathbf{M} \text{ for } \mathbf{x} \in \mathbf{X} \tag{3.2}$$

For completeness, the material *voids* are always included to allow the definition of space exterior to the object's intended boundary. In addition, further restrictions are imposed on the composition function such that the volume fractions always sum to unity and the volume fraction of each material is non-negative:

$$||\mathbf{m}(\mathbf{x})||_1 = \text{material}_0 + \text{material}_1 + \ldots = 1.0 \tag{3.3}$$

$$\text{material}_i \geq 0 \text{ for } 0 \leq i < d_m \tag{3.4}$$

Other than the above constraints, no other limitations have yet been imposed on the nature of the material variation. Tools for working with geometric information (the object's boundary) are still important, but how the material varies throughout the model is now equally important at the time of design. The composition may vary smoothly or contain discontinuities, depending on what tools are available for the definition of $\mathbf{m}(\mathbf{x})$ and how this information is modeled.

It is important to note that at this point in the discussion of FGM modeling issues, the manner in which that data is represented in the computer or how the model is to be processed for fabrication has not been discussed. Only the concept of a Model Space, consisting of a build space and a Material Space, in which an FGM model is to exist have been identified.

3.3 Accuracy in representation

The previous section introduced the goal of FGM modeling: to represent the shape and material variation of an object through a function $\mathbf{m}(\mathbf{x})$. The manner in which $\mathbf{m}(\mathbf{x})$ should be chosen, in part, based on the accuracy of representing the designer's intentions.

3.3.1 Shape

Accuracy in representing shape is an underlying goal of existing CAD systems. The designer has an idea for the shape of an object's boundary. The geometric accuracy is the maximum distance between the desired shape and the shape actually stored in the digital representation (see Figure 3-4).

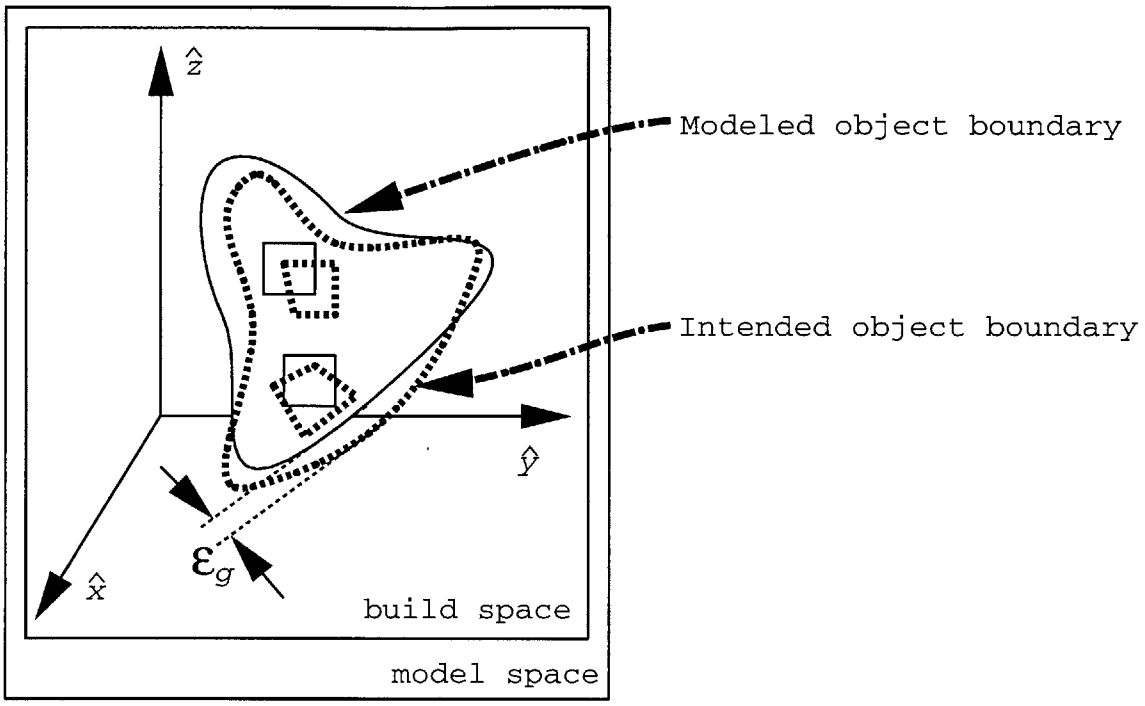


Figure 3-4: The maximum distance between the intended object's boundary and the modeled boundary is the geometric accuracy (ϵ_g) of the modeled object.)

3.3.2 Material

With the representation of graded material information, the concept of material accuracy needs to be defined. The material accuracy is the maximum difference between the intended volume fraction for any material at any point in the object and the volume fraction actually maintained or computed from the digital representation:

$$\begin{aligned} \epsilon_m &= \max\{\mathbf{m}^*(\mathbf{x}) - \mathbf{m}(\mathbf{x})\} \text{ for } \mathbf{x} \in \text{Build Space} \\ &= \max\{|m_0^*(\mathbf{x}) - m_0(\mathbf{x})|, |m_1^*(\mathbf{x}) - m_1(\mathbf{x})|, \ldots\} \text{ for } \mathbf{x} \in \text{Build Space} \end{aligned}$$

where $\mathbf{m}^*(\mathbf{x})$ represents the desired material distribution for the object and $\mathbf{m}(\mathbf{x})$ is the distribution actually modeled in the data structure. Figure 3-5 illustrates the concept of material accuracy for the assignment of uniform material to a region in a model.

3.4 Processing FGM models for fabrication

The issue of processing FGM models for fabrication can now be discussed. With the definition of an FGM object as a function $\mathbf{m}(\mathbf{x})$, the model must be transformed into machine instructions for its

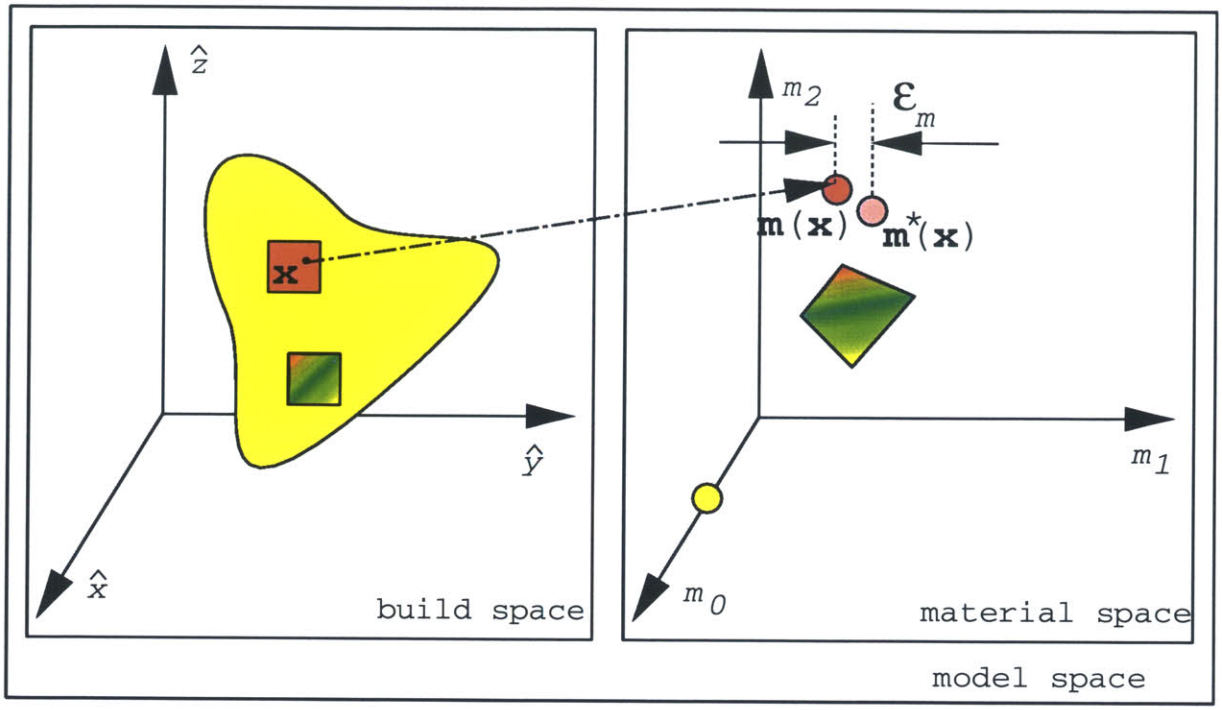


Figure 3-5: Visual interpretation of material accuracy, showing the difference between the desired composition $\mathbf{m}^*(\mathbf{x}_0)$ and the modeled composition $\mathbf{m}(\mathbf{x}_0)$ at the point $\mathbf{x}_0$ in Build Space.

fabrication. To achieve this goal, image processing techniques provide a two-dimensional solution which may readily be adapted for three-dimensional models.

3.4.1 Image processing

Figure 3-6 outlines the steps in image processing, as presented by Ulichney [71, 70]. This process starts with a continuous tone image ($\mathbf{I}_{in}(\mathbf{x})$) that defines the intensity of each of the primary colors over the space occupied by the image.

$$\mathbf{I}_{in}(\mathbf{x}) = \begin{bmatrix} \%red = r(\mathbf{x})) \\ \%green = g(\mathbf{x}) \\ \%blue = b(\mathbf{x}) \end{bmatrix} \text{ for } \mathbf{x} \in \text{ image} \tag{3.5}$$

$\mathbf{I}_{in}(\mathbf{x})$ may originate from a digital photograph, a computer graphics program, or a word processing document.

The first step in image processing is the sampling of the continuous tone image at a grid of

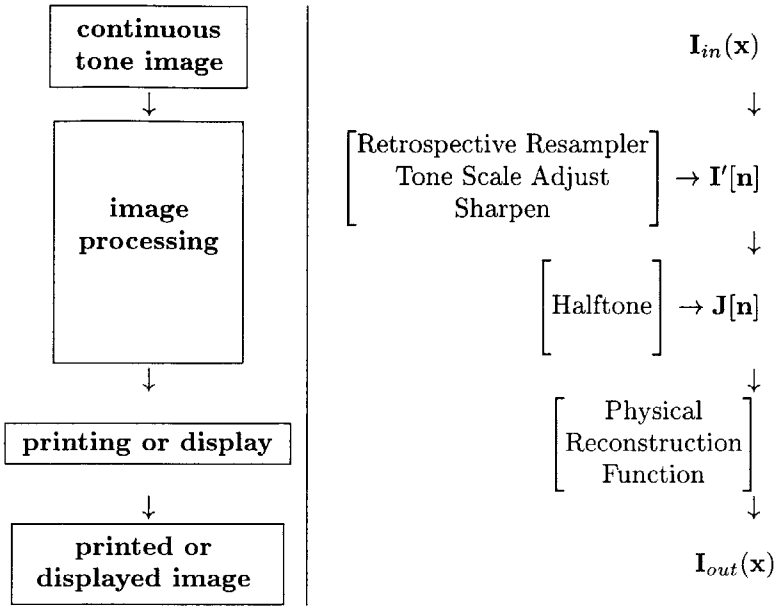


Figure 3-6: Steps of the information flow for image processing.

discrete points ($\mathbf{n}$).

$$\mathbf{I}'(\mathbf{n}) = \begin{bmatrix} \%red = r(\mathbf{n})) \\ \%green = g(\mathbf{n}) \\ \%blue = b(\mathbf{n}) \end{bmatrix} \text{ for } \mathbf{n} \in \text{ grid} \tag{3.6}$$

These sampled values are then modified to compensate for errors inherent in display or printing process. The *Tone Scale Adjust* step modifies values to account for any non-linearities between the mapping of the desired values and the rendered value due to the nature of the Physical Reconstruction Function and the halftoning method used. The *Sharpen* step attempts to modify the values of the image to compensate for any loss of clarity during the halftoning step.

The next step in the information flow is the halftoning of the continuous values in the grid into binary values. There are many methods for accomplishing this, including the use of dithering arrays and error diffusion, but the underlying goal remains the same: determine a pattern of ones and zeros over the sampling grid that will most closely reproduce the original image after the Physical Reconstruction Function is applied.

$$\mathbf{J}(\mathbf{n}) = \begin{bmatrix} 1 \text{ or } 0 \\ 1 \text{ or } 0 \\ 1 \text{ or } 0 \end{bmatrix} \text{ for } \mathbf{n} \in \text{ grid} \tag{3.7}$$

These binary values are encoded into instructions for the display or hardcopy device. The Physical

Reconstruction Function is defined by Ulichney as the mathematical description of this device, capturing the nature of pixel rendering (accuracy and intensity distribution) to reproduce the processed image [70]:

$$\mathbf{I}_{out}(\mathbf{x}) = \begin{bmatrix} \%red = r(\mathbf{x})) \\ \%green = g((x)) \\ \%blue = b((x)) \end{bmatrix} \text{ for } \mathbf{x} \in \text{ image} \tag{3.8}$$

The left hand side of the above equation ($\mathbf{I}_{out}(\mathbf{x})$) represents the final, processed (displayed or printed) image. Although this image was produced through a digital process (pixels on a monitor or ink droplets on a page), ideally this final image will closely match the initial image: $\mathbf{I}_{out}(\mathbf{x}) \approx \mathbf{I}_{in}(\mathbf{x})$ for $\mathbf{x}$ in the image. A measure of the quality of a display device and the image processing algorithms used to drive it, is how close these two images match [70].

This information flow is illustrated in Figure 3-7 for a single color (red). The initial distribution of red in the image is defined as:

$$r(x, y) = \frac{\cos(x) * \sin(y) + 1}{2} \tag{3.9}$$

This distribution is shown in Figure 3-7(a). The image is then sampled over a 77×77 grid, as shown in Figure 3-7(b). The value at each grid point may take on any value between zero and unity. To convert the grid of continuous tone value to binary values, the values are halftoned, generating the grid of binary values shown in Figure 3-7(c). A simple Physical Reconstruction Function based on diffusion of the ink during printing was used to simulate the rendered image Figure 3-7(d). For a more detailed explanation about this process, consult Ulichney's work on digital halftoning [70].

3.4.2 FGM model processing

As previously stated, an FGM object can be defined as the function $\mathbf{m}(\mathbf{x})$, providing a mapping from a Build Space into a Material Space. For the processing of FGM models for fabrication through Local Composition Control, the paradigm of information flow from image processing can be followed. This process is outlined in Figure 3-8.

The process begins with capturing the designer's intent in terms of a digital FGM model. At this point, the model $\mathbf{m}(\mathbf{x})$ is maintained within a data structure selected to accurately capture the designer's ideas.

Once the model is completely defined, processing begins with the model first positioned in the fabrication space, then sampled over a lattice of points, and then halftoned into discrete material primitives. The object is then fabricated through an SFF process, whose properties are captured mathematically by the Physical Reconstruction Function.

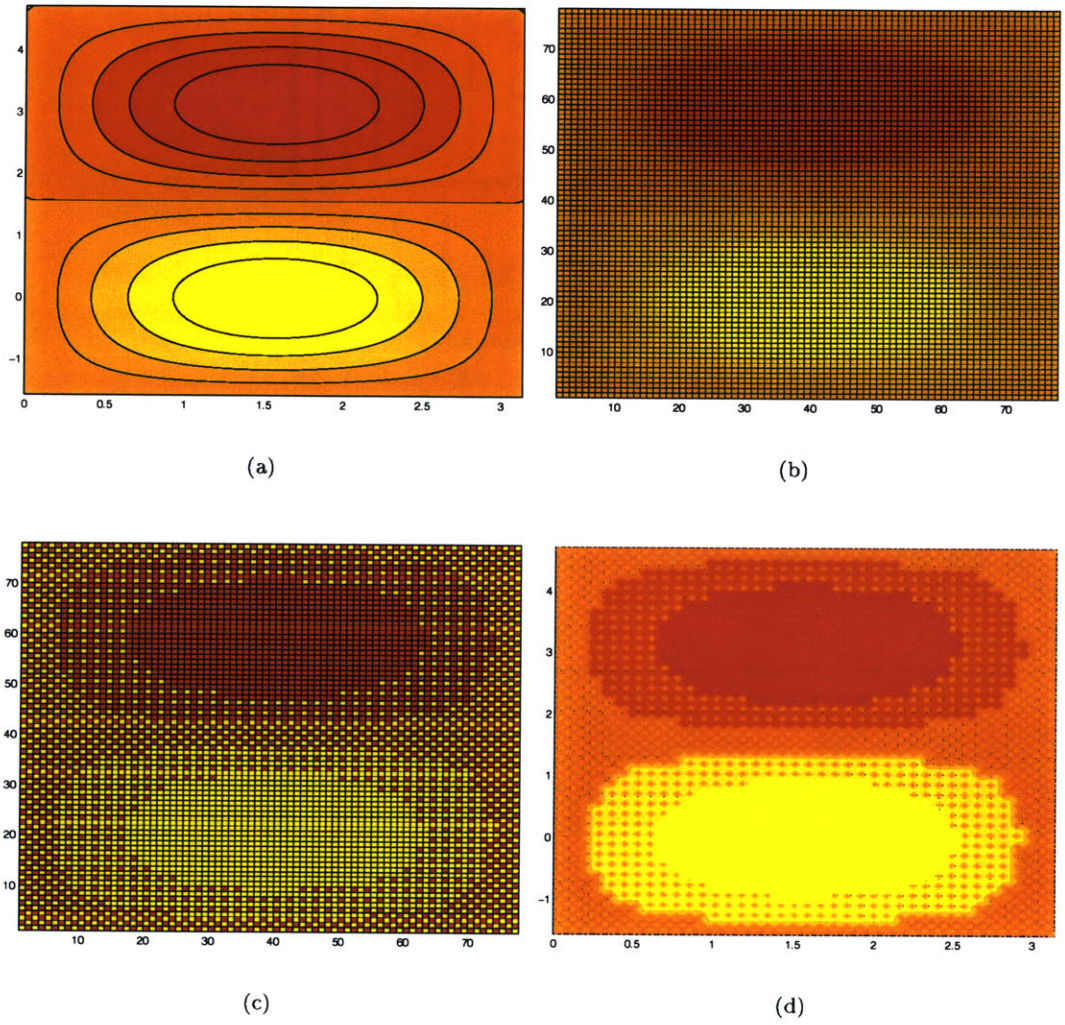


Figure 3-7: (a) Initial image: $\mathbf{I}_{in}(\mathbf{x})$. (b) Sampled image: $\mathbf{I}'(\mathbf{n})$. (c) Halftoned image: $\mathbf{J}(\mathbf{n})$. (d) Physically reconstructed image: $\mathbf{I}_{out}(\mathbf{x})$.

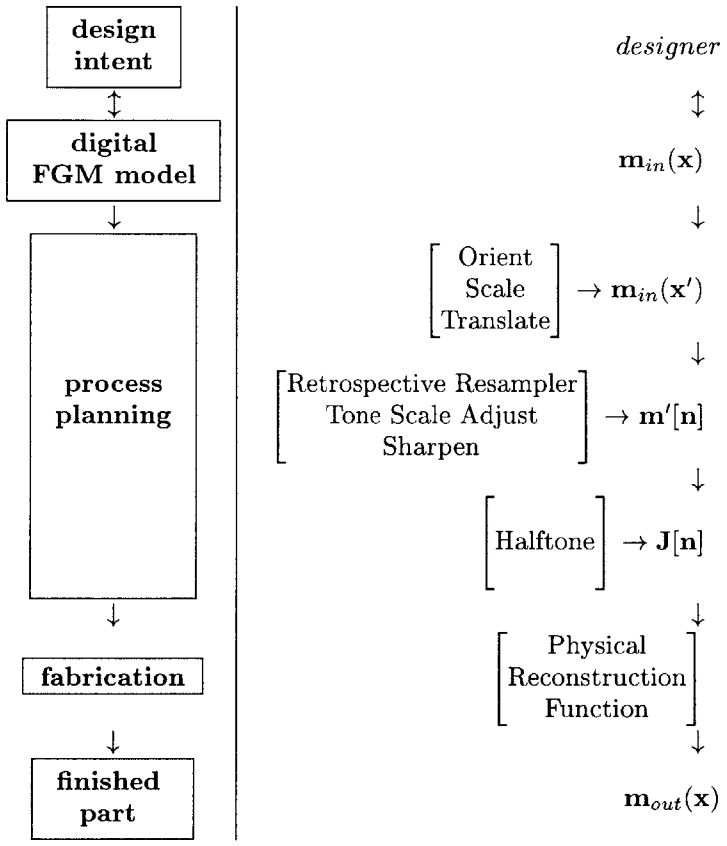


Figure 3-8: Steps of the information flow for FGM model processing.

As an example, consider the definition of an FGM object in a two dimensional Material Space consisting of stainless steel and voids. The processing begins with the part being oriented, scaled, and positioned in the fabrication space in order to ensure an optimal build. The model is then sampled over a lattice of points in the fabrication space. Just as the sampled image needed to be filtered, Tone Scale Adjust and Sharpen steps are applied to help compensate for errors and distortions introduced later. These values are then halftoned into binary values which are then used to drive the fabrication (represented by the Physical Reconstruction Function). For 3D Printing, the Physical Reconstruction Function depends on the nature of the droplet formation, flight, impact, and diffusion into the powder bed. For Selective Laser Sintering, the Physical Reconstruction Function would capture then intensity distribution of the laser and the nature of the melting and solidification of the power.

3.5 Discussion

This chapter has introduced three important concepts: the Model Space in which an FGM object should be defined, the definition of material accuracy, and how FGM models should be processed. This Model Space consists of a Build Space and a Material Space.

Through the introduction of a Model Space, no reference was made to a solid modeling method. Instead, emphasis was placed on the information that needs to be conveyed: shape and material variation. The accuracy in capturing the designer's intent for shape and composition is then defined. Since the goal of any modeling system is to capture design intent, the measures of accuracy (ϵ_g and ϵ_m) should be used as the parameters when evaluating any data structure upon which an FGM modeling system and exchange standard will be built.

Following the paradigm of image processing for the process planning of FGM objects, a clean separation is established between object modeling (part design and representation) and object fabrication. The processing steps are sufficiently general to apply to any point-wise manufacturing process (SLS [6], 3DP [57], SDM [74], SALD [35], SLA [31]) and material system. The details of the halftoning, its encoding into machine instruction, and the nature of the Physical Reconstruction Function, however, are unique to each process and material system and are not addressed in this work [79].

Chapter 4

Modeling composition through decomposition

4.1 Motivation

Solid modeling schemes differ in the manner in which they define interior regions of models, and how these regions are related to each other to form a complete object. In most existing approaches to solid modeling for use in design tasks, attributes attached to regions within the representation of an object are considered to be uniform or constant throughout each region. This permits the association of a single material or manufacturing process with each region, facilitating the modeling of composite structures. With the developing capability to locally vary the composition of a part as it is fabricated, additional information must be associated with each region to represent how its composition varies or grades throughout its interior. This requirement impacts both the complexity of the representation schemes used for each region as well as the number of regions required to accurately model an object. Both of these factors are related to how efficiently the data representing an object data is stored.

To address these issues, this chapter introduces several modeling schemes for representing FGM objects. The data structures to be considered include a exhaustive enumeration approach, a triangulated boundary representation (a variation on the STL format) [47], a finite element mesh [2], the Radial-Edge data structure [73], and the Cell-Tuple-Graph data structure [7, 8, 25, 26]. Regardless of the data structure used, the concept of decomposing the model into regions with which material information is associated is fundamental. The generality in the representation of regions and amount of topological (connectivity) information explicitly recorded, however, is different for each method, resulting in different degrees of generality, accuracy, and efficiency of interrogation algorithms. To illustrate what information is maintained for each method, the object in Figure 4-1 is used. This

chapter is concerned with outlining how the different modeling schemes represent FGM models as well as formulating storage cost of each method in terms of the data it maintains.

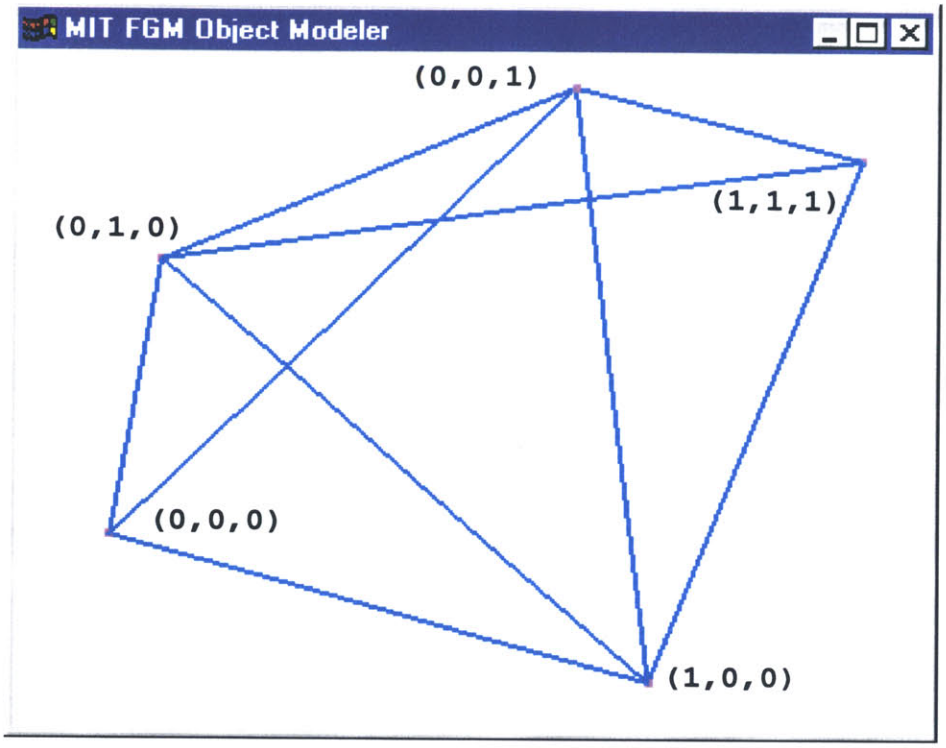


Figure 4-1: Model consisting of two tetrahedra used to illustrate various modeling methods. Shown is the wireframe of the model.

The solid modeling data structures explored here increase in complexity from an exhaustive enumeration scheme to the completely general methods (Radial-Edge and Cell-Tuple-Graph). For each approach, the overall data structure is first introduced, followed by a description of each of the classes needed in an object-oriented implementation of the data structure [54]. The storage requirement for the entire data structure is then formulated in terms of these elementary data classes, providing general expressions for the costs associated with modeling an object within each representation. The analysis here is concerned with the minimal information needed to represent the model and does not include additional information which may be useful for performing operations on the model, such as interrogating the data structure, visualizing or processing the model, and modifying its geometry or composition.

4.2 Nomenclature

The storage costs for the various approaches to modeling FGM objects are derived in terms of the amount and classes of data they maintain and the number of instances of each class stored in the data structure. The cost associated with a specific data class is represented by the symbol S_a with

the subscript a replaced by the name of the corresponding data type or class. The storage cost S_a is the amount of memory required to store a single instance of class a. The units are assumed to be in bytes, although in all but the voxel-based approach the units are arbitrary. The symbols S_{int}, S_{flt}, and S_{ptr} represent the number of bytes for primitive data types integer, floating point number, pointer, respectively, which are machine-dependent.

The number of instances for a given class is represented by the parameter n_b, where b is the class name.

In some cases, attributes within an object are used in the derivation. This information is represented by naming the object followed by a period and the name of the attribute. The number of facets n_f bounding a facetted region r, for instance, would be written as $r.n_f$.

For each modeling method, the terms of the storage costs are initially grouped by data class, identifying the amount and kind of data associated with each data class. This takes the following form:

$$S_{method\ A} = [S_{class\ B}]_{class\ B} + [S_{class\ C}]_{class\ C} \tag{4.1}$$

where the storage cost for modeling method A, class B, and class C are represented by $S_{method\ A}$, $S_{class\ B}$, and $S_{class\ C}$, respectively. In this example, modeling method A consists of only two classes: B and C.

4.3 Voxel-based modeling

In the exhaustive enumeration approach to solid modeling, a region of space containing the object is decomposed into a lattice of voxels [37]. If the model is composed of a single material of uniform density, a binary value is associated with each voxel, indicating whether or not the voxel lies inside or outside of the object. This modeling method, also known as *binary voxel modeling*, is usefule in representing conventional composite structures where discrete materials exist within separate regions of the model. For blended compositions, a vector of numerical values must be associated with each voxel, representing the volume fractions of each material present within the voxel. These models are known as *grey-value voxel models* and are commonly used for modeling sampled data sets of scalar fields.

Figure 4-2 shows the two classes required in a voxelized modeling scheme for representing object. The class Voxel-Model maintains information about the dimensions of each voxel $(\delta_x, \delta_y, \delta_z)$, the number of voxels, or subdivisions, along each dimension (n_x, n_y, n_z), and the position of the lower corner of the voxel model in the Build Space (o_x, o_y, o_z). The pointer *lattice* provides a reference into the 3D grid of voxels and *matsys* contains all of the information regarding the Material System out of which the object is composed. Figure 4-3 shows the sample object represented as a voxel-

based model. Each voxel maintains an array of values $\mathbf{m}$, corresponding to the intensity level or

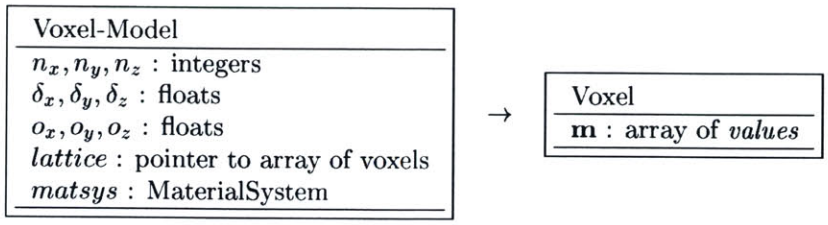
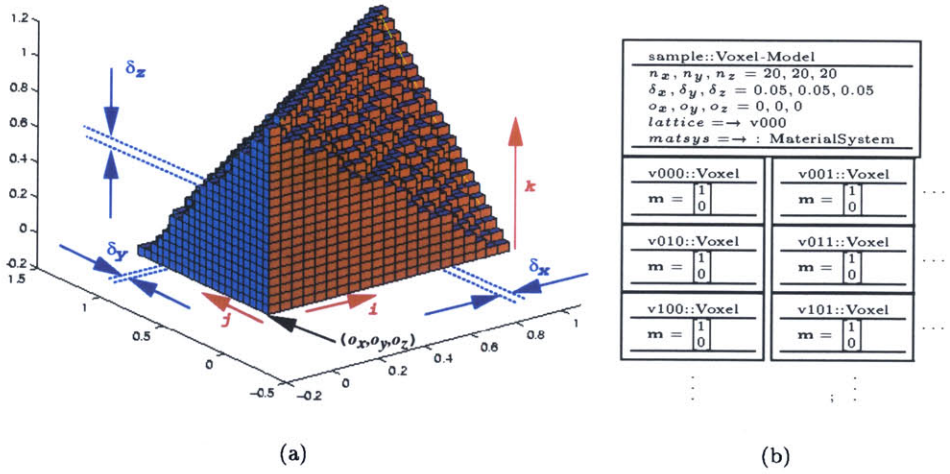


Figure 4-2: Relationships between classes in an exhaustive enumeration method for modeling FGM objects.



(a) (b)

Figure 4-3: (a) Voxelized representation of sample model. (b) VoxelModel object model.

volume-fraction of the corresponding material or physical property. The memory required to store this array depends on the number of levels of intensity (n_λ) to be represented by each voxel as well as the number of fields (materials) in $\mathbf{m}$. To represent n_λ levels of intensity for each material, each voxel must store an integer value from zero to $n_\lambda - 1$ for each material. Representing this value in binary form, the number of bits required by a voxel to represent the intensity value of each material is simply $\lceil \lg n_\lambda \rceil$. For a Material System with d_m materials, each voxel would require $d_m \lceil \lg n_\lambda \rceil$ bits. Therefore, the total storage cost (in bytes) for the exhaustive enumeration representation presented in Figure 4-2 is:

$$S_{vox} = [3S_{int} + 6S_{flt} + S_{ptr} + S_{ms}]_{\text{Voxel-Model}} + \left[\frac{1}{8} n_{vox} d_m \lceil \lg n_\lambda \rceil\right]_{\text{Voxel}} \tag{4.2}$$

where $n_{vox} = n_x n_y n_z$ is the number of voxels stored in the lattice.

Therefore, the storage requirement for the exhaustive enumeration method is directly proportional to the number of voxels (regions) and the log of the number of levels of intensity for each

material stored in the data structure.

4.4 Triangulated shells

The next level of sophistication for modeling FGM objects is the use of a faceted Boundary Representation (B-rep) modeler [47]. In this approach, a model is decomposed into regions of uniform (constant) composition, bounded by shells of triangular facets. This is a modest extension of the conventional STL format in current use for exchanging models for SFF [47]. The data classes

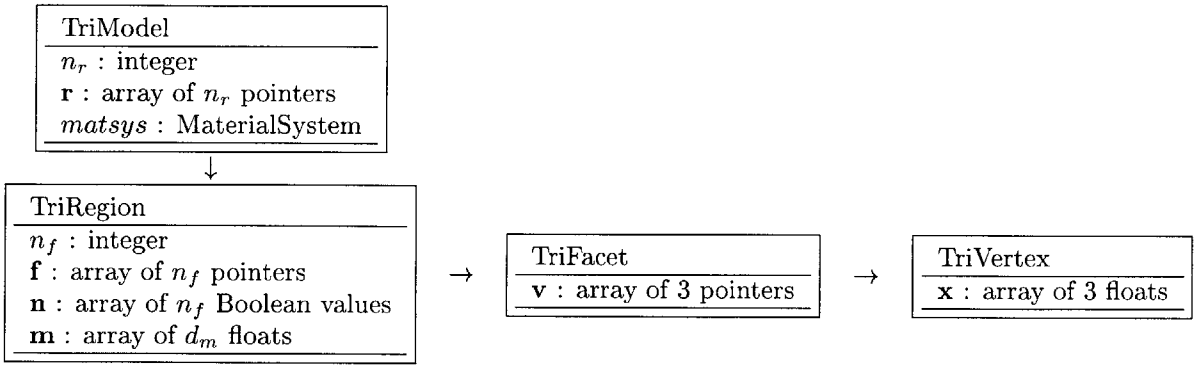
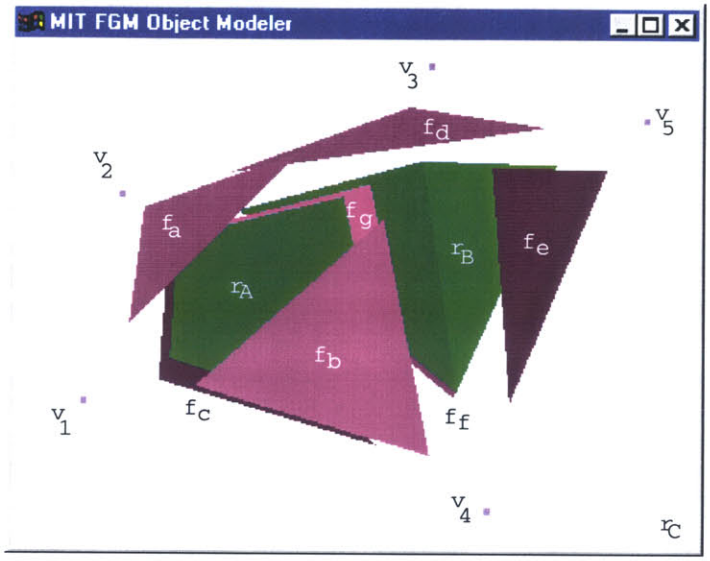


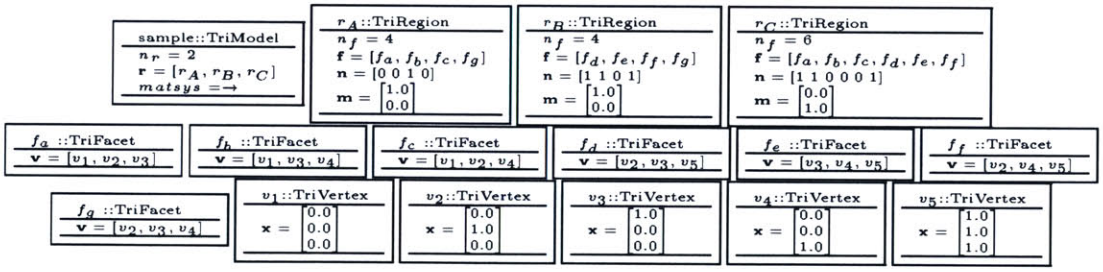
Figure 4-4: Relationships between classes in the triangulated boundary representation approach for modeling FGM objects.

comprising this modeling method include: TriModel, TriRegion, TriFacet, and TriVertex (see Figure 4-4). The top most class, TriModel, is used to represent an individual FGM model, containing a Material System *matsys* and an array of regions **r** into which the model is decomposed. Each region is represented by an object of type TriRegion, containing a vector of volume fractions **m** of the primary materials to define its uniform composition as well as references to the triangular facets **f** defining its boundary. An array of Boolean values **n** is also maintained by each region, each value indicating whether or not the orientation of the vertices about the corresponding triangular facet form a counter-clock-wise circuit when viewed from outside the region. Each triangular facet is an instance of a TriFacet and references three vertices **v**. Vertices are stored separately as TriVertex objects, each storing a point **x** in Build Space. The sample object represented as a triangulated model with its objects are shown in Figure 4-5. The storage cost for the data structure is a function of the number of regions, facets (internal and external), and vertices needed to represent the model.

$$\begin{aligned} S_{tri} &= [S_{int} + n_r S_{ptr} + S_{ms}]_{TriModel} + \left[\sum_{r \in \text{ Model.}\mathbf{r}} (S_{int} + r.n_f S_{ptr} + r.n_f S_{bln} + d_m S_{flt}) \right]_{TriRegion} \\ &\quad + [3n_{triangles} S_{ptr}]_{TriFacet} + [3n_{nodes} S_{flt}]_{TriVertex} \\ &= (S_{int} + S_{ptr} + d_m S_{flt}) n_r + \sum_{r \in \text{Model.}\mathbf{r}} r.n_f \, (S_{ptr} + S_{bln}) + 3 S_{ptr} n_{triangles} + 3 S_{flt} n_{nodes} \\ &\quad + S_{int} + S_{ms} \end{aligned} \tag{4.3}$$



(a)



(b)

Figure 4-5: (a) Triangulated shell representation of sample model. (b) Object model showing the instances data required to represent the sample model in the data structure.

With the region exterior to the modeled object explicitly represented by a TriRegion object, all of the triangles are referenced exactly twice, once by each region incident to it. Each facet is reference twice by definition, defining a facet of the boundary of a single region which is also a facet of the interface between exactly two adjacent region. By representing the complement of the space occupied by the model as a region within the structure, all of the facets (including those on the *external boundary* of the object) are referenced twiced). Therefore,

$$\sum_{r \in \text{Model.}\mathbf{r}} r.n_f = 2n_{triangles} \tag{4.4}$$

where r is a region in the model (stored in the array Model.$\mathbf{r}$), $r.n_f$ are the number of facets bounding the region, and $n_{triangles}$ is the *total* number of triangles stored in the data structure. The above expression for the storage cost simplifies to:

$$\begin{aligned} S_{tri} &= (S_{int} + S_{ptr} + d_m S_{flt})n_r + (5S_{prt} + 2S_{bln})n_{triangles} + 3S_{flt}n_{nodes} \\ &\quad + S_{int} + S_{ms}. \end{aligned} \tag{4.5}$$

4.5 Finite element meshes

In order to facilitate the representation of graded regions (rather than the piece-wise constant regions presented in the preceding two methods), analytic functions defining how the composition varies must be attached to each region. Such a representation can be achieved through the use of a finite element mesh, in which material or physical property fields are attached to the nodes in the mesh. Interpolation functions associated with the volume elements are used to define the composition $\mathbf{m}$ throughout each element as functions of the values assigned to the nodes. Although various finite elements can be defined with interpolation functions of varying degree, for the purposes of this analysis we will restrict the types of elements used in the finite element mesh to linear tetrahedra. The inclusion of additional element definitions would likely reduce the memory costs by some constant factor, but the trends in memory requirements will follow those of homogeneous tetrahedral meshes.

The main classes in a finite element modeling data structure for FGM objects include the FE-Model, FE-Tet, and FE-Vert (see Figure 4-6). An FGM model would consist of a single FE-Model object containing a Material System *matsys* and references to a set of n_r elements $\mathbf{r}$ into which it is decomposed. For the purpose of the memory analysis here, each region within the model is represented by an FE-Tet object, defining a tetrahedral domain of the model and a linear variation of the composition over the domain. Each tetrahedron maintains references to four vertices ($\mathbf{v}$) which are interpolated to define its geometry. The composition information may either be associated with the vertices of the tetrahedra (similar to interpolating nodes in traditional finite element meshes) or

the tetrahedra themselves (see Figures 4-7(a) and (b)). In the former case, FE-Tet objects maintain references to 4 FE-Vert-M objects which are interpolated to define the tetrahedron's geometry and composition. In the latter case, by associating material information **m** with each tetrahedron of class FE-Tet-M, only the geometry is defined by interpolating the referenced FE-Vert objects, permitting the definition of composition independent of neighboring elements and allowing the representation of *discontinuities* in composition between elements (which is a desirable design system feature). Another variation on this modeling approach is the incorporation of backwards references from each vertex to the tetrahedra which are incident to it. This additional topological information (stored in FE-Vert-P and FE-Vert-PM objects) may be useful for increasing the efficiency of some operations on the database, such as evaluating the variation of the geometry or composition across multiple elements and the determination of external (boundary) facets.

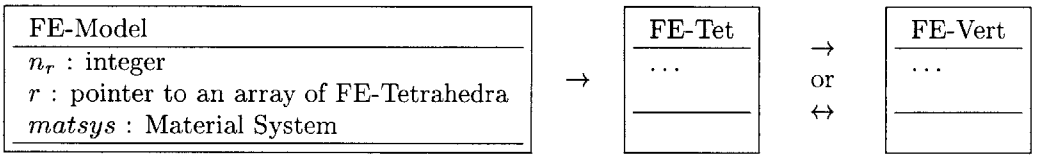
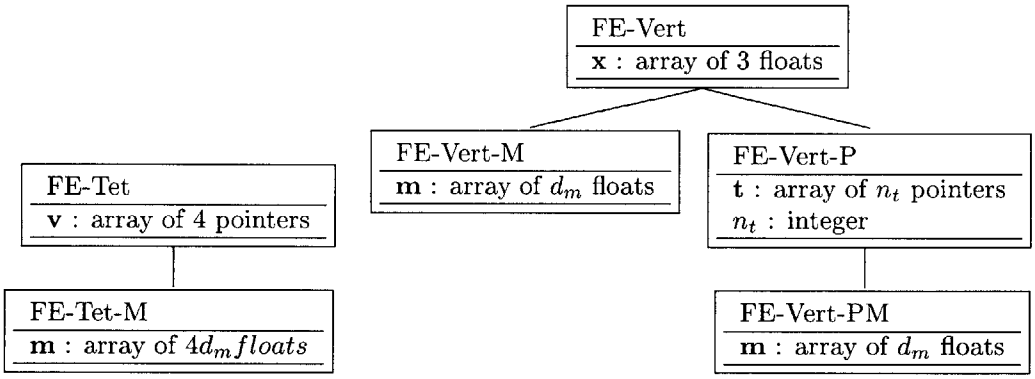


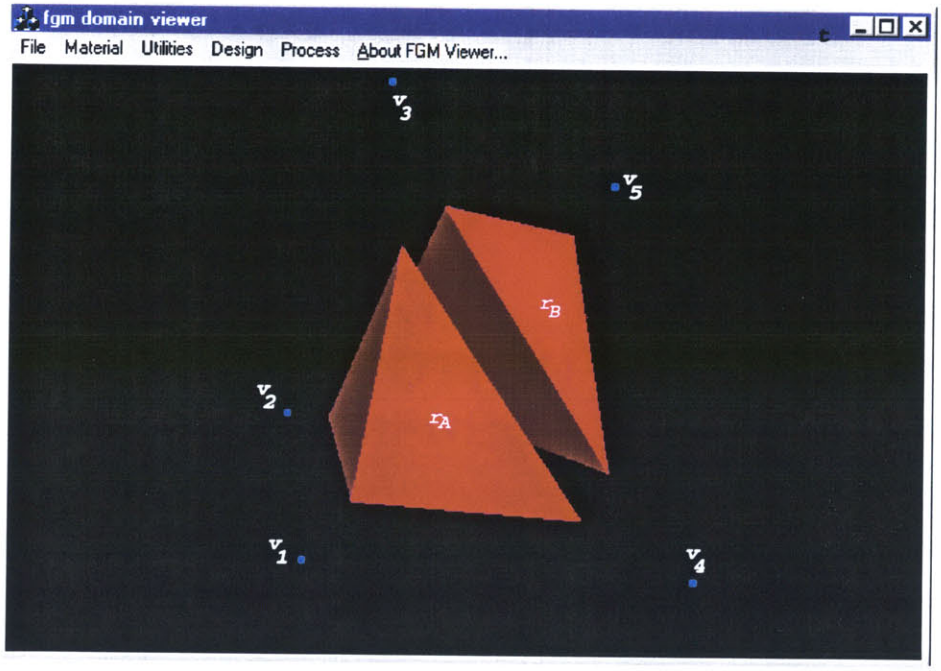
Figure 4-6: Relationships between classes in tetrahedral mesh approach to modeling FGM objects.



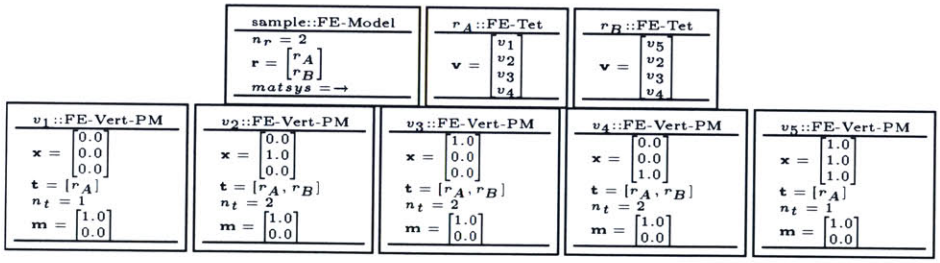
(a) FE-Tet class.

(b) FE-Vert classes.

Figure 4-7: (a) Tetrahedron classes for consideration in meshed modeling representations. (b) Vertex classes for consideration in meshed modeling representations.



(a)



(b)

Figure 4-8: (a) Tetrahedral mesh representation of sample model. (b) Object model showing the instances data required to represent the sample model in the data structure.

$$S_{fe_{00}} = S_{\text{FE-Model}} + S_{\text{FE-Tet}} + [(3+d_m)S_{flt}n_v]_{\text{FE-Vert-M}} \quad (4.6)$$

$$S_{fe_{01}} = S_{\text{FE-Model}} + S_{\text{FE-Tet-M}} + [3S_{flt}n_v]_{\text{FE-Vert}} \quad (4.7)$$

$$S_{fe_{10}} = S_{\text{FE-Model}} + S_{\text{FE-Tet}} + \left[\sum_{v\in\text{FE-Model}} [(3+d_m)S_{flt} + v.n_tS_{ptr} + S_{int}]\right]_{\text{FE-Vert-PM}} \quad (4.8)$$

$$S_{fe_{11}} = S_{\text{FE-Model}} + S_{\text{FE-Tet-M}} + \left[\sum_{v\in\text{FE-Model}} (3S_{flt} + v.n_tS_{ptr} + S_{int})\right]_{\text{FE-Vert-P}} \quad (4.9)$$

where $S_{\text{FE-Model}} = S_{int} + S_{ptr} + S_{ms}$, $S_{\text{FE-Tet}} = 4n_rS_{ptr}$,

and $S_{\text{FE-Tet-M}} = 4(S_{ptr} + d_mS_{flt})n_r$.

Equations 4.6- 4.9 detail the storage costs for the four variations of the finite element approach to modeling FGM objects. In Equations 4.6 and 4.8, the material information is associated with the vertices and interpolated by the tetrahedron's interpolation functions, whereas in Equations 4.7 and 4.9 each tetrahedron maintains its own composition information independently. In the first two equations, references are maintained only by the tetrahedra to the vertices. For the cases in which references are also maintained from the vertices back to their incident tetrahedra, represented in the Equations 4.8 and 4.9, the storage cost is given in terms of the number of tetrahedra incident to each vertex. Depending upon the topology of the model, therefore, the storage requirement for each vertex will be different. The total storage cost of the model, however, can be determined by recognizing that the total number of incidence relationships within the data structure from vertices to tetrahedra is equal to the number of incidence relationships from tetrahedra to vertices. Given that each tetrahedron references exactly 4 vertices, the number of references from the vertices to the tetrahedra can be expressed in terms of the number of tetrahedra in the model:

$$\#\text{ of vertex-to-tetrahedron relationships: } \sum_{v\in\text{ Model}} v.n_t = \sum_{r\in\text{Model}} r.n_{vertices} = \sum_{j=0}^{n_r-1} 4 = 4n_r \quad (4.10)$$

where v is a vertex in the Model, $v.n_t$ is the number of tetrahedra incident to the vertex v, r is a tetrahedron in the Model, $r.n_{vertices}$ is the number of vertices incident to the tetrahedron r, and n_r is the total number of tetrahedra in the Model.

With this information, the storage cost for the various tetrahedral data structures can be sim-

plified, as in Equations 4.11- 4.14.

$$S_{te_{00}} = 4S_{ptr}n_{tetrahedra} + (3 + d_m)S_{flt}n_v + S_{int} + S_{ms} + S_{ptr} \quad (4.11)$$

$$S_{te_{01}} = 4(S_{ptr} + d_m S_{flt})n_{tetrahedra} + 3S_{flt}n_v + S_{int} + S_{ms} + S_{ptr} \quad (4.12)$$

$$S_{te_{10}} = 8S_{ptr}n_{tetrahedra} + [(3 + d_m)S_{flt} + S_{int}]n_v + S_{int} + S_{ms} + S_{ptr} \quad (4.13)$$

$$S_{te_{11}} = 4(2S_{ptr} + d_m S_{flt})n_{tetrahedra} + [3S_{flt} + S_{int}]n_v + S_{int} + S_{ms} + S_{ptr} \quad (4.14)$$

Whereas Equations 4.6- 4.9 are equally applicable to a finite element mesh with any type of linear volume element (by replacing $S_{\text{FE-Tet}}$ and $S_{\text{FE-Tet-M}}$ with the elements' corresponding storage costs), Equations 4.11- 4.14 apply to only the subset of tetrahedral meshes.

4.6 Generalized cellular decomposition or multi-region B-rep

The preceding sections introduced three methods for representing FGM objects. They are considered *specialized* methods here since they restrict the decomposition of an object into regions (voxels, triangulated shells, or tetrahedra). In current practice, however, CAD systems provide a wider range of representations to precisely describe the boundary surfaces of solid models. This is achieved through the use of generalized data structures which maintain the topology of a model in a relational database. This allows the incorporation of various geometric representations that best describe the geometry of the object's model. This paradigm can be extended to the representation of FGM objects, permitting the representation of models decomposed into regions of arbitrary topology.

Two possible data structures for achieving a generalized representation are presented in this section. Both maintain the topology of a model in terms of a graph and reference definitions for the geometry and composition of each topological entity external to the topological data structure. The first method is known as the Radial-Edge data structure [73] and represents the basis for exchange standards of 3D object models such as STEP [28] and IGES [64], and is widely adopted as the modeling kernel within various solid modeling systems, such as ACIS [65]. The second solid modeling method is the Cell-Tuple-Graph data structure [7, 8, 25, 26, 32] which can be considered as a generalization and simplification of the Radial-Edge data structure for n-dimensional models. Other generalized data structures for modeling solid models exist [1, 23, 56] but are not discussed here since the methods choosen here reflect the general nature of these other methods and the same trends should apply.

For both modeling methods, the geometry and composition is defined external to the topological data structure, allowing a modular approach to the design of the FGM modeling system architecture. For the purpose of FGM representation, the concept of an *FGMDomain* is introduced, representing a generic structure through which the geometry and material fraction variation is defined for

the referencing topological entity of any dimension. The purpose of this structure is to map the corresponding topological entity into build and Material Spaces, uniquely defining some part of an FGM model. This mapping is subject only to the constraints that it is defined over the topological entity's interior and provides a one-to-one mapping into Build Space, guaranteeing that the domain does not self-intersect.

The remainder of this section is divided into two parts, describing these concepts in more detail. The first part introduces the frameworks of the two modeling methods and how they maintain the connectivity between topological entities. The memory cost for each of these methods is defined in terms of their fundamental components. The second part of the section describes the concept of FGMDomains in more detail and provides the definition for a set of domains which could be used to represent a wide class of FGM parts, and identifies the storage cost of each domain class. In the presentation of these data structures, material information is associated with all FGMDomain classes.

Although material information *may* not be needed at the lower dimensions (points, curves, and surfaces), this information is associated with these classes in this analysis for three reasons: consistency, unambiguous representation of composition, and flexibility for future development. The concept of an FGMDomain is generic, as will be described in the following sections, and by associating material information with an FGMDomain, all FGMDomains are handled equally. In addition, in order to provide an unambiguous definition of the composition at each point in Build Space, material information associated with lower dimensional entities allows the unique definition of the composition at points at interfaces between adjacent regions.[1] Finally, there is the possibility of developing new FGMDomains for which the composition is derived from lower dimensional entities (a mesh interpolating material information at nodes is one example) or defining design tools that perform operations to create compositions over higher dimensional entities from the compositions associated with the lower dimensional ones (such as lofting) . By including this information at the lower dimensions in this analysis, the conclusions drawn will still apply to future work that may require information at these levels.

4.6.1 Data structures for topology

Radial-Edge data structure

Overview of Radial-Edge Classes: The Radial-Edge data structure provides a unified method for representing solid models [73]. The data structure maintains the topology of the models in terms of two major sets of classes: (1) topological entities and (2) their uses. The former set of classes

[1] Another approach to handling the composition definition over interfaces is the use of half-open regions in which compositions at points on a face are derived from one of the incident regions, determined from the orientation of the face.

represent the different topological entities of the model and include the entities Vertex, Edge, Loop, Face, Shell, and Region. The second set of classes simplifies the implementation of the modeling system architecture, providing information about how instances of the first set of classes are used. Each instance of an object of type Vertexuse, Edgeuse, Loopuse, or Faceuse identifies a single role that the corresponding topological entity plays in the connectivity of the model. The hierarchy of the all the classes in the Radial-Edge data structure is shown in Figure 4-9. The following sections will describe in detail the role of each class in the data structure and the data each maintains.

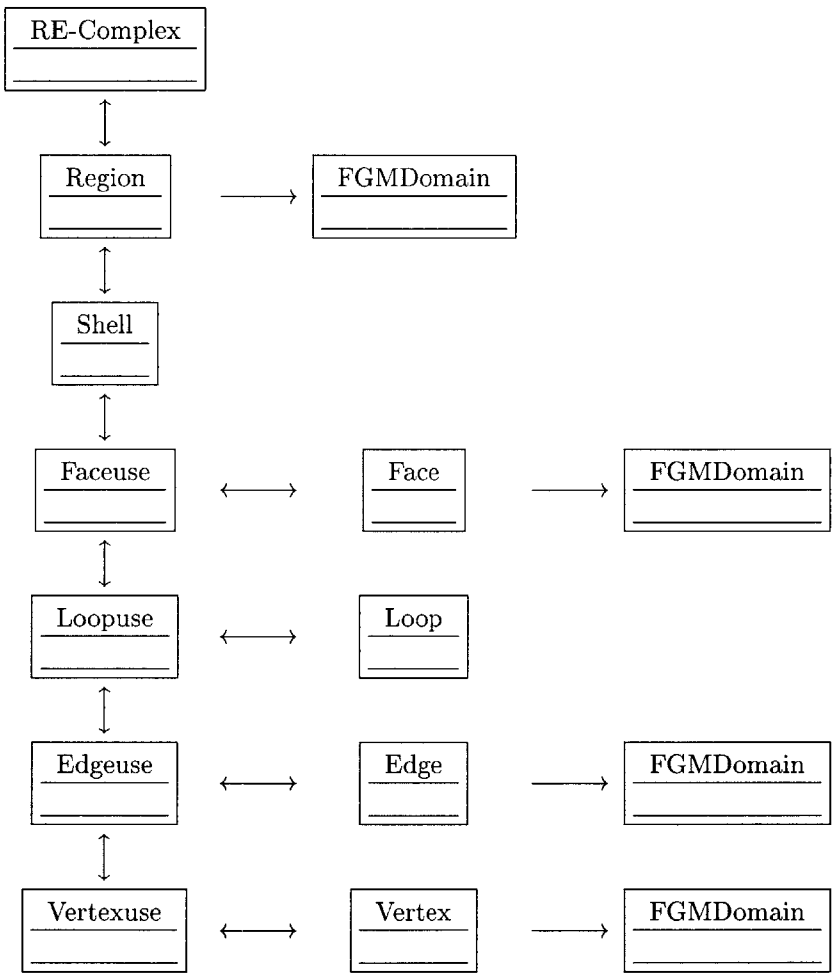
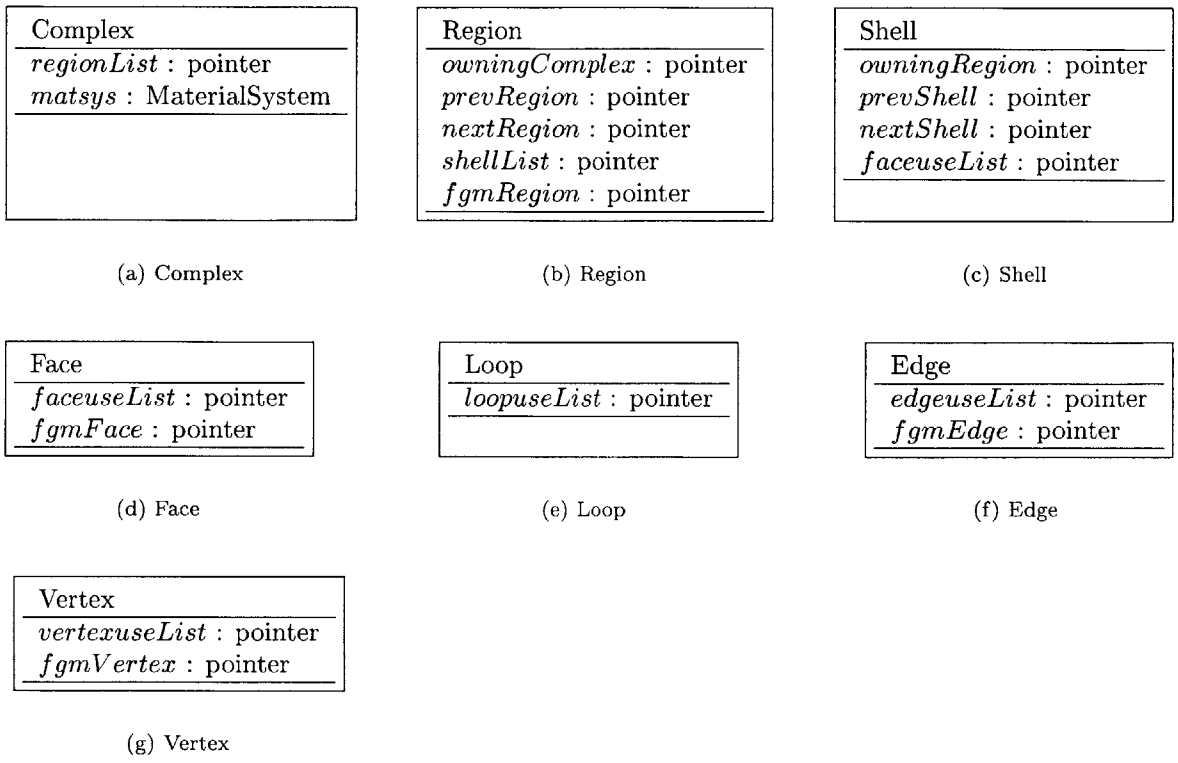


Figure 4-9: The classes comprising the Radial-Edge data structure and how they are related.

Topological classes: In the adaptation of the Radial-Edge data structure for FGM modeling, each FGM object is represented by an instance of the Complex class, serving as the access point to the database. The Complex node maintains the MaterialSystem (*matsys*) out of which the model is created as well as a reference (*regionList*) into a circular list of Regions. The shape and composition of the Complex are defined by the data stored in the lower dimensional topological objects, beginning with the referenced Regions objects. In the traditional Radial-Edge data structure, references to



(a) Complex (b) Region (c) Shell

(d) Face (e) Loop (f) Edge

(g) Vertex

Figure 4-10: Topological classes defined within the Radial-Edge data structure and their attributes.

other Complexes are maintained within this class, creating a circular list of Complexes stored in the data structure. Since this information is not directly relevant to the memory analysis, and in the interest of brevity, it is not included here.

Region objects within the Radial-Edge data structure represent sub-regions or volumes of space within a Complex, bounded by Shells of Faces. The topology of a Region object is arbitrary, subject to the requirements that at most one Region within the Complex is infinite in extent and all Regions are bounded by Shells of Faces. A Region node in the data structure maintains an upward reference to the Complex (*owningComplex*) to which it belongs and a downwards reference into a list of shells (*shellList*) which define its boundary. Each Region node also maintains horizontal references to the previous region (*prevRegion*) and the next region (*nextRegion*) contained in the model (these regions may or may not be adjacent). In this way, all of the Regions in a single Complex are linked together in a circular list. To define a region's shape and composition, the Region node finally references an FGMDomain object of dimension 3 (*fgmRegion*), defining the geometry and material grading over the region's interior.

The boundary of a Region is defined by a list of Shells. Each Shell represents an oriented surface serving as a boundary to its referenced Region (*owningRegion*). A Region may be bounded by multiple shells, permitting the representation of voids and disjoint regions. To provide this flexibility,

the Shells are maintained in a circular list fashion, with each Shell referencing the previous Shell (*prevShell*) and the next Shell (*nextShell*) bounding the region. The final piece of data maintained by a Shell node is a reference into a list of Faceuses ($faceuseList$) which define the geometry and orientation of the shell. No composition or geometric information is explicitly associated with the Shell class. It is used purely to maintain the relationships between topological entities of dimensions 3 and 2 (Regions and Faces).

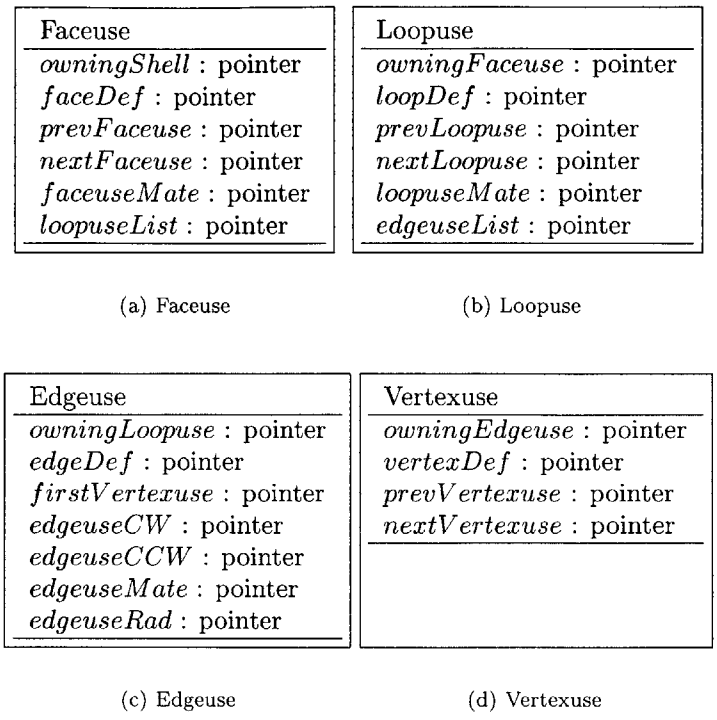
Each Shell is composed of a set of Faces, each of which represents a bounded section of the Shell. A Face is a topological entity of dimension 2 and is orientable, with each side bounding a different Region. The orientation of a Face is not maintained explicitly within its definition since orientation is relative. For each Region to which a Face is incident, the Face has a different orientation. This information is captured by the Faceuse objects stored in the list referenced by $faceuseList$. To define the geometry and composition of the Face, the Face node also references an FGMDomain of dimension 2 ($fgmFace$).

Just as Regions are bounded by Shells, Faces are bounded by Loops. A Loop consists of an alternating sequence of Vertices and Edges, forming a circuit. Each boundary of a Face is represented by a single Loop, and as such Loops are also orientable. The Edges and Vertices comprising a loop are not explicitly maintained within a Loop node since the orientation depends on its use, but are referenced indirectly through a list of Loopuses (*loopuseList*).

The Vertices and Edges comprising Loops represent the lower dimensional (0 and 1) entities defining the model. Each Vertex or Edge may be referenced by multiple Loops, yielding multiple uses for a given Vertex or Edge within a model (as maintained by the lists referenced by *vertexuseList* and *edgeuseList*, respectively). As with Faces and Regions, the definition of a Vertex or Edge is maintained by an externally referenced FGMDomain of the appropriate dimension ($fgmVertex$ or $fgmEdge$).

Topological uses: As stated above, each topological entity within a model may have multiple roles within the relational database. A Face, for instance, serves as part of two different Shells bounding two adjacent Regions. Similarly, an Edge may be a part of any number of Loops bounding Faces incident to the Edge. To define all of the roles for the various topological entities, four additional classes are introduced. An instance of each object represents one role the corresponding topological entity plays in the connectivity of the parts in the entire Complex. These classes (shown in Figure 4-11) include: Faceuse, Loopuse, Edgeuse, and Vertexuse.

Each Face in the model serves a portion of two different Shells, defining the interface between two adjacent regions. To represent this information, a Faceuse is defined for each orientation of the Face. Each Faceuse maintains a reference to the Shell (*owningShell*) to which it belongs as well as to the definition of the Face ($faceDef$). All of the Faceuses defining the orientation of the



(a) Faceuse (b) Loopuse

(c) Edgeuse (d) Vertexuse

Figure 4-11: The Uses of topological entities within the Radial-Edge data structure.

owning Shell are linked together in a circular list, with each of its Faceuses referencing a previous Faceuse (*prevFaceuse*) and next Faceuse (*nextFaceuse*) in the shell. In addition, since each Face has exactly two uses (one for each region to which it is adjacent), each Faceuse maintains a reference to its opposite mate (*faceuseMate*). Finally, to define the oriented boundary of the Face according to the orientation of the Face relative to the Region, a list of Loopuses is referenced by a Faceuse (*loopuseList*) for each Loop bounding the Face.

As previously described, the boundary of each Face is defined by one or more Loops. Each Loop has two possible orientations relative to the orientation of the Face (clockwise or counter-clockwise), requiring the orientation of the Loop to be defined for each orientation of the Face. To maintain this information, the class Loopuse is introduced. Each Loopuse maintains a reference to the Faceuse (*owningFaceuse*) for which it is maintaining the orientation of the boundary. Since a Face may be bounded by multiple Loops (to represent interior holes), the Loopuses for a given Faceuse are linked together in a circular list with each Loopuse referencing the previous (*prevLoopuse*) and the next (*nextLoopuse*) Loopuse for the owning Faceuse. In addition, each Loopuse also has a mate (*loopuseMate*) oriented in the opposite direction, representing the same boundary of the Face on its opposite side. Since a Loopuse represents one orientation of a Loop, each Loopuse references a list of Edgeuses (*edgeuseList*), each of which maintain a correctly oriented Edge contained in the Loopuse.

Each Edge in a model has two endpoints (A and B), yielding two possible directions along which the Edge may be traversed ($A \rightarrow B$ or $B \rightarrow A$). To distinguish between the two directions, the class Edgeuse is introduced and is used to represent each instance a given Edge is used in a Loop. Each Edgeuse references the Loopuse (*owningLoopuse*) to which it belongs as well as the definition of its Edge (*edgeDef*). The Vertex at the start of the Edge of the given Edgeuse's orientation is also referenced (*firstVertexuse*), thereby defining the orientation of the Edge for the owning Loopuse. To maintain the order of the Edges in the Loopuse, each Edgeuse references then next Edgeuses in the clockwise orientation (*edgeuseCW*) and counter-clockwise (*edgeuseCCW*) directions about the Loopuse, permitting the circuit of Edges for a given orientation of a Face to be followed. Each Edgeuse also maintains references to the opposing Edgeuse (*edgeuseMate*) on the opposite of the owning Loopuse's Face and to the Edgeuse on the radially adjacent Face (*edgeuseRad*) to the owning Loopuse's Face.

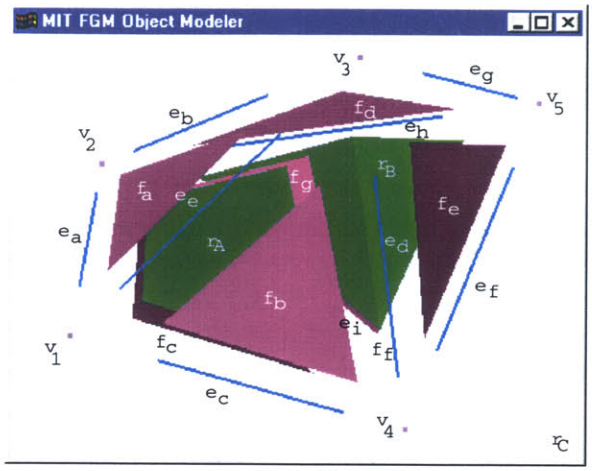
The final class, Vertexuse, is used to represent each instance a Vertex is used within a Complex. Each Vertexuse represents the role of the referenced Vertex (*vertexDef*) as the starting point for the directed the traversal of an Edge, as defined by the owning Edgeuse (*owningEdgeuse*). In addition, all of the possible uses for the referenced Vertex are linked in a circular list, with each Vertexuse referencing the previous (*prevVertexuse*) and next (*nextVertexuse*) Vertexuses for the Vertex.

Storage costs associated with representing topology with the Radial-Edge data structure: The previously described data classes are used to completely define the topology of a solid model (or how the topological elements defining the model are connected together) within the Radial-Edge data structure. Referring to Figures 4-10 and 4-11, the cost for each instance of each class within the modeling method is readily determined, as summarized in Table 4.1.

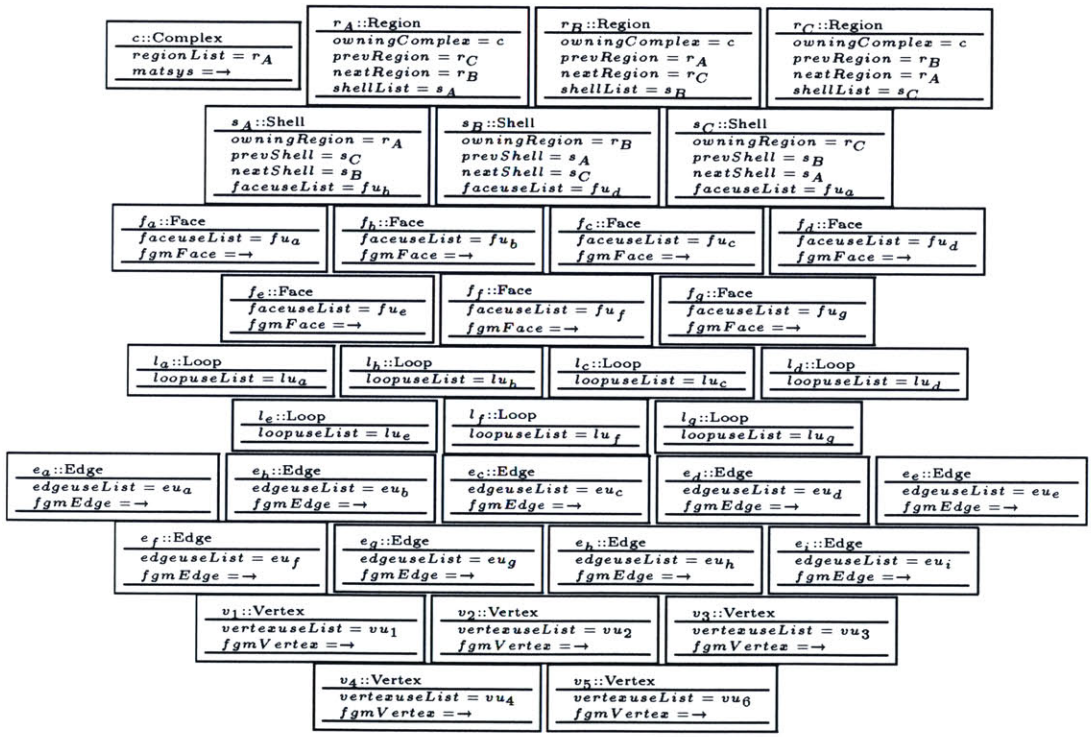
Class	Storage cost
Complex	$S_{ptr} + S_{ms}$
Region	$5S_{ptr}$
Shell	$4S_{ptr}$
Face	$2S_{ptr}$
Loop	S_{ptr}
Edge	$2S_{ptr}$
Vertex	$2S_{ptr}$
FaceUse	$6S_{ptr}$
LoopUse	$6S_{ptr}$
EdgeUse	$7S_{ptr}$
VertexUse	$4S_{ptr}$

Table 4.1: Storage costs for each instance of each class within the Radial-Edge data structure.

The total storage cost attributed to representing the topology of a model maintained in the



(a)



(b)

Figure 4-12: (a) Radial Edge representation of sample model. (b) Object model showing the instances of topological entities required to represent the sample model in the Radial Edgedata structure.

Radial-Edge data structure as a function of the number objects of each class is expressed as [2]

$$\begin{aligned} S_{re} &= S_{ptr} + S_{ms} + 5S_{ptr}n_r + 4S_{ptr}n_s + 2S_{ptr}n_f + S_{ptr}n_l + 2S_{ptr}n_e + 2S_{ptr}n_v + \\ &\quad 6S_{ptr}n_{fu} + 6S_{ptr}n_{lu} + 7S_{ptr}n_{eu} + 4S_{ptr}n_{vu} + \sum_{fgmd \in model} S_{fgmd}. \end{aligned} \tag{4.15}$$

The storage cost associated with the definition of the object's shape and composition is represented by the last term, $\sum_{fgmd \in model} S_{fgmd}$, which depends on the type and nature of the FGMDomains used to model the object. Several possible FGMDomains are introduced in Section 4.6.2 along with their storage costs.

Recognizing that each Face has only two Faceuses (one for each Region to which it is adjacent), the above equation simplifies to:

$$\begin{aligned} S_{re} &= S_{ptr} + S_{ms} + 5S_{ptr}n_r + 4S_{ptr}n_s + 14S_{ptr}n_f + S_{ptr}n_l + 2S_{ptr}n_e + 2S_{ptr}n_v + \\ &\quad 6S_{ptr}n_{lu} + 7S_{ptr}n_{eu} + 4S_{ptr}n_{vu} + \sum_{fgmd \in model} S_{fgmd}. \end{aligned} \tag{4.16}$$

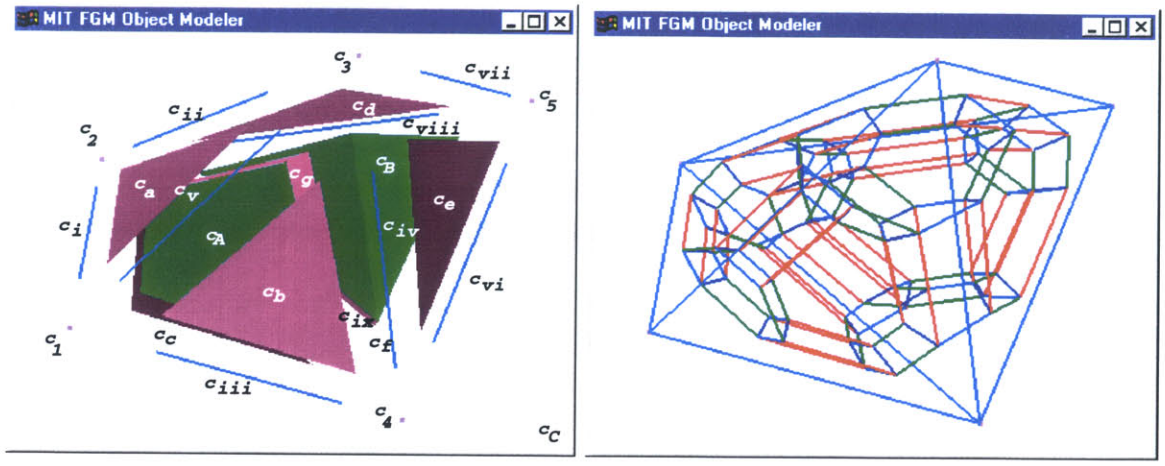
Cell-Tuple-Graph

The Cell-Tuple-Graph data structure [7, 8, 25, 26, 32] is a cellular solid modeling method, decomposing a model of an object into *Cells*. A Cell is a generic, topological entity of any dimension, representing a vertex, edge, face, or region in the model. In this way, this data structure can be considered as a generalization of the Radial-Edge data structure, providing a uniform framework for representing topology and relating the topological entities to each other.

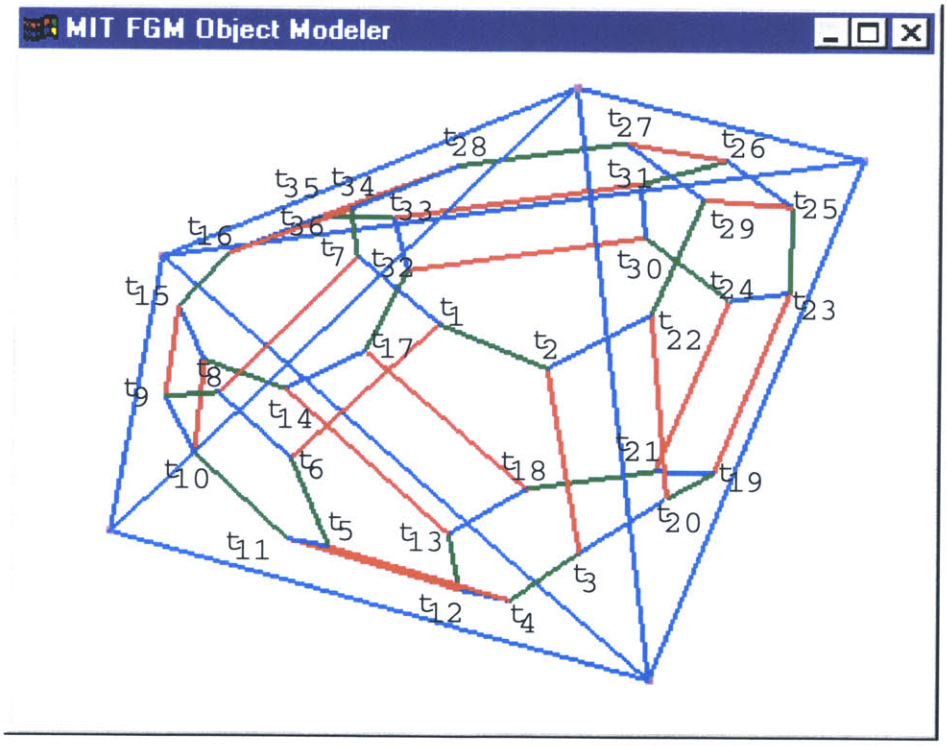
There are two approaches to implementing the Cell-Tuple-Graph data structure. The first approach is to maintain an incidence graph between Cells, in which each Cell maintains lists of incident Cells of higher and lower dimensions. Information about adjacent Cells is determined by intersecting lists of incident Cells to find which Cells share a common boundary. The second approach is to implement the Cell-Tuple-Graph in a fashion analogous to the Radial-Edge structure. For this case, connectivity between the Cells is explicitly maintained in graphs of Tuples which reference Cells. In this way, adjacency information is explicit and neighboring Cells can be found in constant time. Due to the fact that it is analogous to the Radial-Edge structure and provides faster access to searching the data structure, only the latter approach is discussed here.

The Cell-Tuple-Graph data structure is composed of only four classes: CTG (Cell-Tuple-Graph), Cell, Tuple, and FGMDomain. Cells represent the topological entities in the model, FGMDomains describe their shape and composition, and instances of the Tuple and CTG classes are used to maintain the connectivity between Cells. The relationships between these classes are shown in

[2]The number of instances of Regions, Shells, Faces, Loops, Edges, Vertices, Faceuses, Loopuses, Edgeuses, and Vertexuses in a Radial-Edge model are represented by n_r, n_s, n_f,n_l, n_e, n_v, n_{fu}, n_{lu} , n_{eu}, and n_{vu}, respectively.



(a) (b)



(c)

Figure 4-13: (a) Cells in model. (b) Graph of tuples. Paths between tuples are colored according to dimension: red=0, green = 1, blue = 2, dashed blue/yellow = 3. (c) Graph of tuples over boundary with each tuple labelled.

Figure 4-14. More detailed descriptions of the data maintained by each of the classes and their roles in the Cell-Tuple-Graph modeling method are provided below.

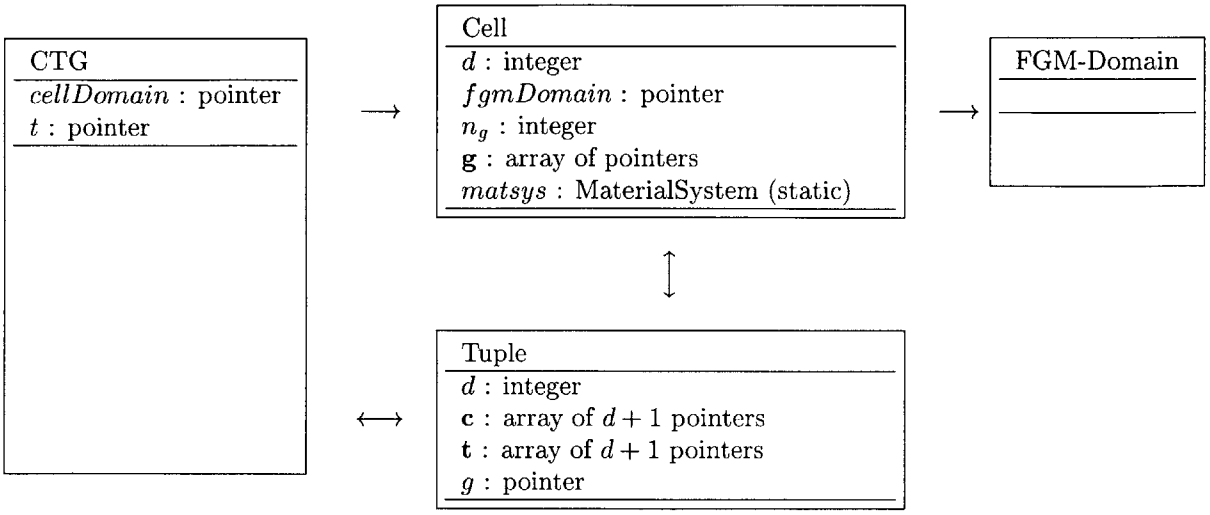


Figure 4-14: Relationships between classes in Cell-Tuple-Graph data structure.

Cell: A Cell represents a d dimensional topological entity in the model. In reference to the Radial-Edge data structure, a Cell may play the role of Vertex, Edge, Face, or Region within the model. The shape and composition of the Cell is defined by an FGMDomain referenced by the Cell (fgmDomain), allowing a separation between the implementation of the topological data structure and the geometric and material representation. Cells are connected together through graphs of Tuples (described below). Each Cell maintains a single reference to a Tuple in each unique CTG in which the Cell appears. References into the n_g graphs containing the Cell are maintained in an array of Tuples (**g**) within the Cell object. A MaterialSystem is also defined to represent the material space in which the model exists. Since the Material Space must be the same for all of the Cells in the model, the allocation of only one MaterialSystem (*matsys*) is required per model. At the time of model creation, a Cell of dimension 3 is defined to represent the Build Space. It is at this time that the MaterialSystem is defined for the Cell class, establishing a Material Space for the Cells that will define the object.

Tuple: A Tuple represents a unique collection of incident Cells (**c**) in a model of differing dimensions: $\mathbf{c} = \{c_0, c_1, \ldots c_d\}$. Each Cell in a Tuple represents a section of the boundary of the next higher dimensional Cell within the same Tuple: $c_k \subset \partial c_{k+1}$. Each Tuple is connected to $d+1$ other Tuples within the reference CTG (g) by d numbered paths.

CTG: A CTG consists of a collection (graph) of Tuple objects connecting all of the incident states which differ by a single Cell together. Just as the Tuple's role is to provide the incidence information

between Cells of different dimensions, a CTG object provides the adjacency information between Cells of the same dimensions, thereby allowing complete representation of a subdivided, manifold section of a model.

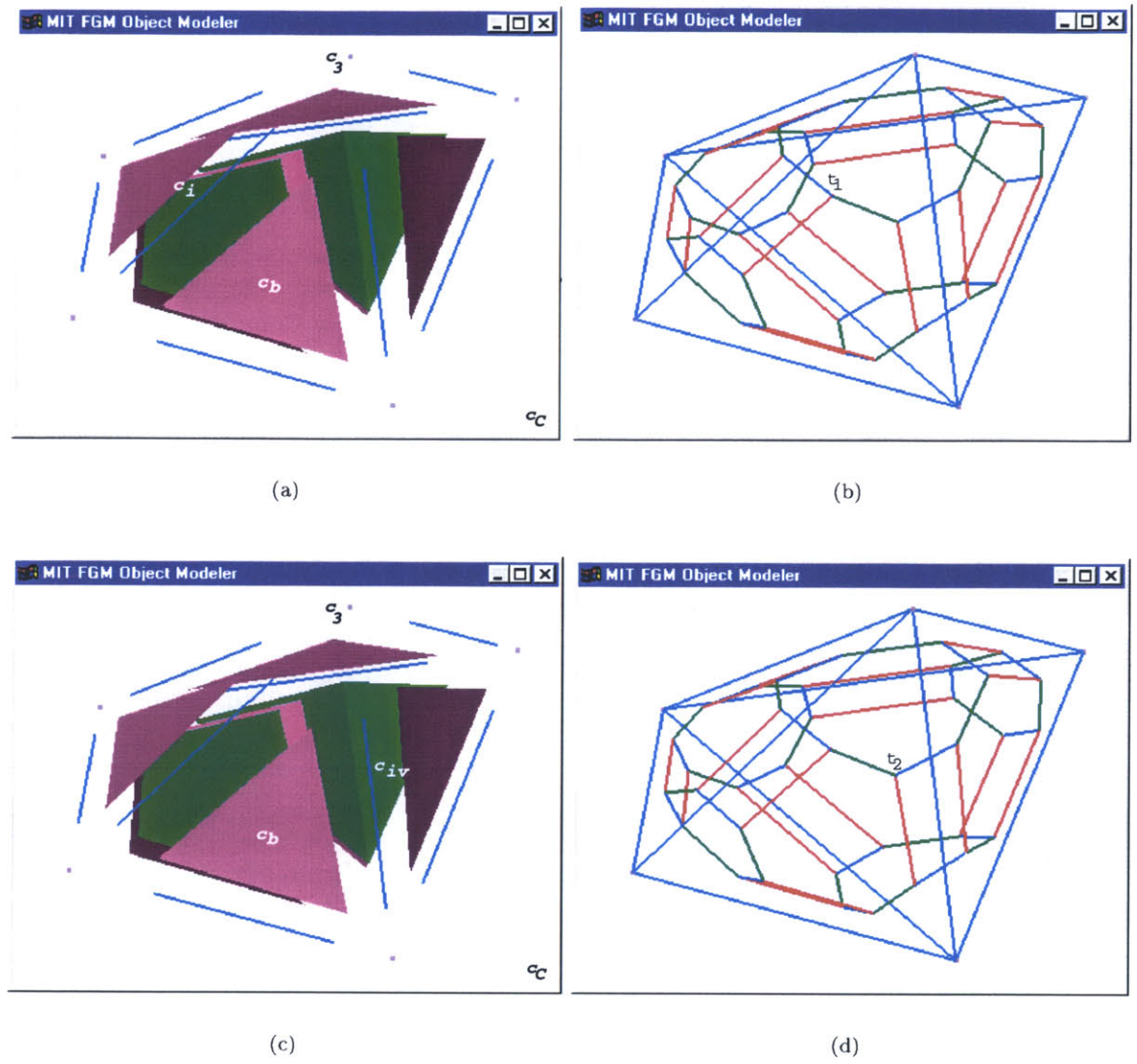


Figure 4-15: Illustration of switch operator performed on tuple t_1 for dimension 1 (green). (a) Cells in tuple t_1. (b) Location of tuple t_1 in graph. (c) Cells in Tuple t_2. (d) Location of tuple t_2 in graph.

All of the Tuples of a single CTG object can be accessed through the repeated application of a single operator: *switch* . Given Tuple t_α and a dimension k, switch(k, t_α) returns the Tuple $t_\beta = t_\alpha.t_k$. By definition of the CTG, Tuples t_α and t_β reference all of the same Cells except at dimension k. These two Cells ($t_\alpha.c_k$ and $t_\beta.c_k$) are adjacent to each other (for $k > 0$) and as well incident to the same Cells of lower and higher dimension in the model. Since all of the Tuples for a single graph can be reached through this single operator, a CTG object needs only to reference a

single Tuple.

A CTG also maintains a reference to a Cell (*cellDomain*) in which the subdivided manifold exists. All of the Cells referenced by the graph are said to exist within the domain of this higher dimensional Cell. To provide access between CTGs, the *jmpswitch* operator is defined. Given a Tuple t_α and a Cell c_β, jmpswitch(t_α, c_β) returns a Tuple t_γ in a different graph (if one exists) than the one referenced by t_α ($t_\alpha.g \neq t_\gamma.g$). The new Tuple will either contain the Cell c_β within its array of referenced cells ($c_\beta \in t_\gamma.\mathbf{c}$) or the new Tuple will serve as a node in a graph defined within Cell c_β ($t_\gamma.g.cellDomain = c_\beta$). With the addition of this operator, the CTG data structure can represent Faces with internal Loops and Regions with internal Shells in a manner analogous to the Radial-Edge data structure.

FGMDomain: To define the shape and composition of each Cell, the FGMDomain class is used. The definition of a derived FGMDomain class is arbitrairy, subject to the only restriction that it span the interior of the Cell referencing it, uniquely defining the composition of each point within the Cell. A more detailed description of the FGMDomain class is given in Section 4.6.2.

Class	Storage cost
CTG	$2S_{ptr}$
Tuple	$S_{int} + (2d+3)S_{ptr}$
Cell	$2S_{int} + (\mathbf{c}_\kappa.n_g + 1)S_{ptr}$ [$+S_{ms}$ static]
FGMDomain	[depends on definition]

Table 4.2: Storage costs for different objects within the Cell-Tuple Data structure.

Storage required for a model maintained in the Cell-Tuple-Graph data structure:

$$S_{ctg} = 2S_{ptr}n_{ctg} + \sum_{t \in model}[S_{int} + (2t.d+3)S_{ptr}] + \sum_{c \in model}[2S_{int} + (c.n_g+1)S_{ptr}] + S_{ms} + \sum_{fgmd \in model} S_{fgmd} \quad (4.17)$$

where t represents a Tuple in the Cell-Tuple-Graph, c represents a Cell in the model, an $fgmd$ is an FGMDomain referenced by a Cell. The storage cost an FGMDomain is represented by S_{fgmd} and is depedendent upon the defition of the domain. The following section (Section 4.6.2 introduces several possible FGMDomain definitions for use in modeling FGM objects with a relational database and establishes their storage costs.

4.6.2 FGMDomains

In most B-rep approaches to solid modeling, definitions for the shapes of curves and surfaces defining the boundary of regions are defined external to the topological data structure. This not only simplifies issues in implementation but permits the expansion of the modeling system as new shape

representations are introduced in the future. The same paradigm can be followed to model FGM objects. The concept of an FGMDomain is introduced here as an abstract class to define shape and composition of a point, curve, surface, or region. It is a generic concept and can be used to define FGM models within either the Radial-Edge or the Cell-Tuple-Graph data structures. As previously stated, this approach is not new but is a direct extension of how B-rep modelers currently represent shape. In such applications, a wide range of definitions have been established to provide the flexibility and accuracy needed to represent a wide range of models. Some shape definitions within the STEP standard (a file format for used for the neutral exchange of solid models) were given in Figure 2-4(a) to illustrate the various possibilities for defining shape.

This section presents a set FGMDomains based on rational Bézier formulations which would provide the capability to represent FGM objects with non-linear geometries and compositions. Although the definition of FGMDomains is certainly not limited to the few introduced here, it is anticipated that this limited set will enable the representation of a broad range of FGM objects with complex shapes and compositions. Other possible definitions for FGMDomains are described at the end of this section to re-enforce the generality of the approach and how it can be expanded in the future.

Zero Dimensional FGMDomain

FGMPoint:FGMDomain
x : array of 3 floats
m : array of d_m floats

(a) FGMPoint class.

Figure 4-16: Zero dimensional FGMDomain.

FGMPoint: The lowest dimensional FGMDomain is the FGMPoint. It defines a single point in Build Space (**x**) as well as a point in Material Space (**m**). The storage cost for an FGMPoint, as defined in Figure 4-16, is simply:

$$S_{pnt} = (3 + d_m)S_{flt} \tag{4.18}$$

One Dimensional FGMDomain

FGM Rational Bézier Curve: Although a variety of representations for lines, arcs, and freeform curves could be considered, this analysis will only be concerned with rational Bézier curves for defining one dimensional FGM entities [19, 52, 55]. An FGM rational Bézier curve is a parametric curve which maps a line in parametric space ($0 \leq t \leq 1$) to a rational, freeform curve in the build

$(t \rightarrow \mathbf{x}(t))$ and Material Spaces $(t \rightarrow \mathbf{m}(t))$. In order to be well defined, the geometric mapping must be one-to-one, without self intersections. Due to its general definition, an FGM rational Bézier curve can be used to represent straight line segments as well as polynomial and rational curves, enabling the representation of a wide range of curves within a model.

FGMRationalBézierCrv:FGMDomain
n_x : integer
n_m : integer
$\mathbf{x}$: array of $3 \times (n_x + 1)$ floats
$\mathbf{m}$: array of $d_m \times (n_m + 1)$ floats
$\mathbf{w}_x$: array of $(n_x + 1$ floats
$\mathbf{w}_m$: array of $(n_m + 1)$ floats

(a) FGMRationalBézierCrv class.

Figure 4-17: One dimensional FGMDomain: FGMRationalBézierCrv.

The definition of FGMRationalBézierCurve is given in Figure 4-17(a). Each instance of the curve maintains its degree of shape (n_x) and composition variation (n_m). The shape and composition of the curve are defined by control polygons and weights in Build Space $(\mathbf{x}, \mathbf{w}_x)$ and Material Space $(\mathbf{m}, \mathbf{w}_m)$, respectively. The mapping from parameter space into model space (using inhomogeneous coordinates) is provided by the following pair of equations:

$$\mathbf{x}(t) = \frac{\sum_{i=0}^{n_x} w_{xi}\mathbf{x}_i B_i^{n_x}(t)}{\sum_{i=0}^{n_x} w_{xi} B_i^{n_x}(t)} \qquad \mathbf{m}(t) = \frac{\sum_{i=0}^{n_m} w_{mi}\mathbf{m}_i B_i^{n_m}(t)}{\sum_{i=0}^{n_m} w_{mi} B_i^{n_m}(t)} \tag{4.19}$$

where Bernstein polynomial basis of degree n on the unit interval $t \in [0, 1]$ are defined by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{(n-i)} \tag{4.20}$$

The binomial coefficients are given by

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} \text{ if } 0 \leq i \leq n \\ 0 \text{ otherwise} \end{cases}$$

The memory required to represent an FGMRationalBézierCurve is a function of the degrees of the mapping functions as well as the dimension of the Material Space.

$$S_{rbc} = 2S_{int} + [4(n_x + 1) + (d_m + 1)(n_m + 1)]S_{flt} \tag{4.21}$$

Two Dimensional FGMDomains

Three classes of FGMDomains are introduced here to represent the shape and composition of topological entities of dimension two. The first two are parametric, based on rational Bézier formulations [19, 52, 55]. The third is a more general representation for planar surface, with the topology of surface's boundary implicitly defined by an accompanying topological data base.

FGMRationalBézierTri:FGMDomain
n_x : integer
n_m : integer
$\mathbf{x}$: array of $\frac{3}{2}(n_x+1)(n_x+2)$ floats
$\mathbf{m}$: array of $\frac{d_m}{2}(n_m+1)(n_m+2)$ floats
$\mathbf{w}_x$: array of $\frac{1}{2}(n_x+1)(n_x+2)$ floats
$\mathbf{w}_m$: array of $\frac{1}{2}(n_m+1)(n_m+2)$ floats

(a) FGMRationalBézierTri class.

FGMRationalBézierQuad : FGMDomain
m_x : integer
m_m : integer
n_x : integer
n_m : integer
$\mathbf{x}$: array of $3(m_x+1)(n_x+1)$ floats
$\mathbf{m}$: array of $d_m(m_m+1)(n_m+1)$ floats
$\mathbf{w}_x$: array of $(m_x+1)(n_x+1)$ floats
$\mathbf{w}_m$: array of $(m_m+1)(n_m+1)$ floats

(b) FGMRationalBézierQuad class.

FGMPlanarSurface : FGMDomain
f : pointer
$\hat{\mathbf{n}}$: array of 3 floats
d : float
$\mathbf{m}$: array of d_m floats

(c) FGMPlanarSurface class.

Figure 4-18: Two dimensional FGMDomains: FGMRationalBézierTri, FGMRationalBézierQuad, and FGMPlanarSurface.

FGM Rational Bézier Triangular Patch: The FGMRationalBézierTri class is used to defined the shape and composition of a two dimensional topological entity bounded by three edges through the mapping of a triangular parameter space into the build and Material Spaces. As with the curve case, the mapping into Build Space must be one-to-one in order to prevent self-intersections.

The data maintained by the FGMRationalBézierTri class include the degrees of the shape (n_x) and composition (n_m) variation as well as the control points and weights to define the shape ($\mathbf{x}, \mathbf{w}_x$) and composition ($\mathbf{m}, \mathbf{w}_m$). The mapping from parameter space into build and Material Spaces (using inhomogeneous coordinates):

$$\mathbf{x}(\mathbf{u}) = \frac{\sum_{|\mathbf{i}|=n_x} w_{x\mathbf{i}}\mathbf{x}_\mathbf{i} B_\mathbf{i}^{n_x}(\mathbf{u})}{\sum_{|\mathbf{i}|=n_x} w_{x\mathbf{i}} B_\mathbf{i}^{n_x}(\mathbf{u})} \text{ and } \mathbf{m}(\mathbf{u}) = \frac{\sum_{|\mathbf{i}|=n_m} w_{m\mathbf{i}}\mathbf{m}_\mathbf{i} B_\mathbf{i}^{n_m}(\mathbf{u})}{\sum_{|\mathbf{i}|=n_m} w_{m\mathbf{i}} B_\mathbf{i}^{n_m}(\mathbf{u})}. \tag{4.22}$$

where $\mathbf{i}$ represents the index of a control point or composition and $|\mathbf{i}| = i_0 + i_1 + i_2$. The parameter

$\mathbf{u}$ is the barycentric coordinate of the point in parameter space satisfying the condition $|\mathbf{u}| = u_0 + u_1 + u_2 = 1.0$ and $0 \leq u_0, u_1, u_2 \leq 1$. The function $B^n_{\mathbf{i}}(\mathbf{u})$ is the generalized Bernstein polynomials of degree n, as defined by

$$B^n_{\mathbf{i}}(\mathbf{u}) = \begin{cases} \frac{n!}{i_0! i_1! i_2!} u_0^{i_0} u_1^{i_1} u_2^{i_2} \text{ if } i_0, i_1, i_2 >= 0 \text{ and } i_0 + i_1 + i_2 = n \\ 0 \text{ othewise.} \end{cases} \tag{4.23}$$

Since the influence of each control point and weight are summed over $|\mathbf{i}| = n$, the number of control points and weights required to completely defined the mapping for a Bézier triangle of degree n is:

$$\sum_{|\mathbf{i}|=n} 1 = \sum_{i=1}^{n+1} i = \frac{1}{2}(n+1)(n+2). \tag{4.24}$$

With the information, along with the data contained with each class instance (see Figure 4-18(a)), the storage cost associated with a single FGMRationalBézierTri patch object is determined as:

$$S_{rbtp} = 2S_{int} + \left[2(n_x+1)(n_x+2) + \frac{d_m+1}{2}(n_m+1)(n_m+2)\right] S_{flt}. \tag{4.25}$$

FGM Rational Bézier Quadrilateral Patch: An FGMRationalBézierQuad also defines the shape and composition for a two dimensional topological entity, except that an FGMRationalBézierQuad is bounded by four edges, homeomorphic to a rectangle defined in parameter space. The data maintain by objects of this class include the degrees of shape (m_x, n_x) and composition (m_m, n_m) variation in the $\hat{\mathbf{u}}_0$ and $\hat{\mathbf{u}}_1$ directions in parametric space. The shape and composition are defined by nets of control points in Build Space $(\mathbf{x}, \mathbf{w}_x)$ and Material Space $(\mathbf{m}, \mathbf{w}_m)$. The mapping from parameter space into model space is accomplished through the following pair of equations:

$$\begin{aligned} \mathbf{x}(u_0, u_1) &= \frac{\sum_{i_0=0}^{m_x} \sum_{i_1=0}^{n_x} w_{x i_0 i_1} \mathbf{x}_{i_0 i_1} B^{m_x}_{i_0}(u_0) B^{n_x}_{i_1}(u_1)}{\sum_{i_0=0}^{m_x} \sum_{i_1=0}^{n_x} w_{x i_0 i_1} B^{m_x}_{i_0}(u_0) B^{n_x}_{i_1}(u_1)} \text{ and} \\ \mathbf{m}(u_0, u_1) &= \frac{\sum_{i_0=0}^{m_m} \sum_{i_1=0}^{n_m} w_{m i_0 i_1} \mathbf{m}_{i_0 i_1} B^{m_m}_{i_0}(u_0) B^{n_m}_{i_1}(u_1)}{\sum_{i_0=0}^{m_m} \sum_{i_1=0}^{n_m} w_{m i_0 i_1} B^{m_m}_{i_0}(u_0) B^{n_m}_{i_1}(u_1)}. \end{aligned} \tag{4.26}$$

where (i_0, i_1) is the index of a control point or composition and (u_0, u_1) is a point in parameter space. The function $B^n_i(u)$ is the i^{th} *ordinary* Bernstein polynomial of degree n, as defined in Equation 4.20. Accounting for the data maintained by the class defined in Figure 4-18(b), the storage cost for an FGMRationalBézierQuad patch is:

$$S_{rbqp} = 4S_{int} + [4(m_x+1)(n_x+1) + (d_m+1)(m_m+1)(n_m+1)] S_{flt}. \tag{4.27}$$

General 2D Cell or B-rep Face: If a surface is planar and maintains a uniform composition, information about the shape and composition variation needed not be explicitly defined. A single composition (vector of volume fractions) can be associated with an FGMDomain along with the equation for the plane and the topology of the planar face can be inferred from the topological data structure. To accomplish this, the FGMPlanarSurface is introduced. This class maintains an equation for the normal of a plane ($\mathbf{n}$), its distance from the origin of the Build Space (d), and a vector of volume fractions ($\mathbf{m}$) defining the composition over the plane. Furthermore, the bounding edges of the planar surface are determined through the interrogation of the accompanying topological data structure, accessed through the reference to the corresponding Face or Cell (f). The geometry of the face is defined by the points that lie on the plane ($\hat{\mathbf{n}} \cdot \mathbf{x} = d$) and within the boundary as defined its loops of edge. Through this class, complex, planar surfaces with arbitrary boundaries and internal holes can be defined without the duplication of data maintained by lower dimensional topological elements or the subdivision of the surface into simple facets. The storage cost of each FGMPlanarSurface object is simply:

$$S_{ps} = S_{ptr} + (4 + d_m)S_{flt}. \tag{4.28}$$

Three Dimensional FGMDomains

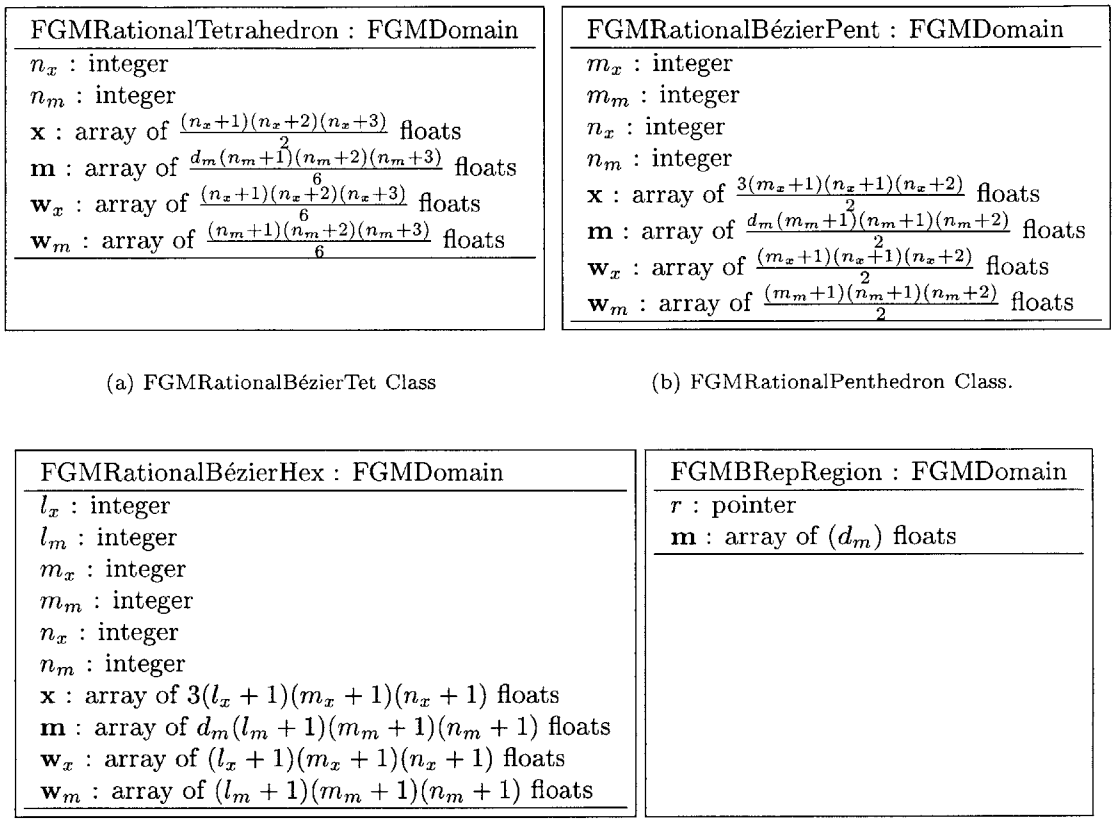
Three dimensional FGMDomains define the shape and composition of a region with an FGM object. Four derived FGMDomains classes are discussed here: the FGMRationalBézierTet, FGMRational-BézierPent, FGMRationalBézierHex, and the FGMBRepRegion. The first three of these class are based on Bézier volume representations [24].

FGM Rational Bézier Tetrahedral Region: The first FGM region class to be discussed is the FGMRationalBézierTet. Instances of this class provide a mapping from a tetrahedral parametric domain into a three dimensional Build Space and the model space according to the following pair equations:

$$\mathbf{x}(\mathbf{u}) = \frac{\sum_{|\mathbf{i}|=n_x} w_{x\mathbf{i}}\mathbf{x}_\mathbf{i} B_\mathbf{i}^{n_x}(\mathbf{u})}{\sum_{|\mathbf{i}|=n_x} w_{x\mathbf{i}} B_\mathbf{i}^{n_x}(\mathbf{u})} \text{ and } \mathbf{m}(\mathbf{u}) = \frac{\sum_{|\mathbf{i}|=n_m} w_{m\mathbf{i}}\mathbf{m}_\mathbf{i} B_\mathbf{i}^{n_m}(\mathbf{u})}{\sum_{|\mathbf{i}|=n_m} w_{m\mathbf{i}} B_\mathbf{i}^{n_m}(\mathbf{u})}. \tag{4.29}$$

As for the one and two dimensional classes, the shape and composition are defined by sets for control points and weights in Build $(\mathbf{x}, \mathbf{w}_x)$ and Material $(\mathbf{m}, \mathbf{w}_m)$ Spaces. The control points and weights are blended using the generalized Bernstein polynomials, as defined by

$$B_\mathbf{i}^n(\mathbf{u}) = \frac{n!}{i_0! i_1! i_2! i_3!} u_0^{i_0} u_1^{i_1} u_2^{i_2} u_3^{i_3} \tag{4.30}$$



(a) FGMRationalBézierTet Class

(b) FGMRationalPenthedron Class.

(c) FGMRationalBézierHex Class

(d) FGMBRepRegion Class

Figure 4-19: Three dimensional FGMDomains: FGMRationalBézierTet, FGMRationalBézierPent, FGMRationalBézierHex, and FGMBRepRegion.

where $\mathbf{u} = (u_0, u_1, u_2, u_3)$ is the barycentric coordinate of the point in a tetrahedral parameter space satisfying the conditions $|\mathbf{u}| = u_0 + u_1 + u_2 + u_3 = 1$ and $0 \leq u_0, u_1, u_2, u3 \leq 1$. $\mathbf{i} = (i_0, i_1, i_2, i_3)$ is the index of a control point or weight. The indices of the control points and weights satisfy the condition $|\mathbf{i}| = i_0 + i_1 + i_2 + i_3 = n$, where n is the degree of the generalized Bernstein polynomial.

For a Bézier tetrahedron of a given degree n , the number of control points or weights defining its shape or composition is

$$\sum_{|\mathbf{i}|=n} 1 = \sum_{i=1}^{n+1} \sum_{j=1}^{i} j = \sum_{i=1}^{n+1} \frac{1}{2} i(i+1) = \frac{1}{6}(n+1)(n+2)(n+3). \tag{4.31}$$

Therefore, the total storage cost for each FGMRationalBézierTet region object is

$$S_{rbtr} = 2S_{int} + \left[\frac{2}{3}(n_x+1)(n_x+2)(n_x+3) + \frac{d_m+1}{6}(n_m+1)(n_m+2)(n_m+3)\right] S_{flt}. \tag{4.32}$$

FGM Rational Bézier Pentahedron Region: The next FGMDomain to be discussed in the FGMRationalBézierPent. Instances of this class are homeomorphic to a topological entity having five faces, nine edges, and six vertices, also known as a wedge. The blending of the control points in Build $(\mathbf{x}, \mathbf{w}_x)$ and Material $(\mathbf{m}, \mathbf{w}_m)$ Spaces to map a wedge in parametric space to a region in model space are accomplished through a combination of the ordinary and generalized Bernstein polynomials. This mapping is given by:

$$\begin{aligned} \mathbf{x}(\mathbf{u}; v) &= \frac{\sum_{j=0}^{m_x} \sum_{|\mathbf{i}|=n_x} w_{x\mathbf{i};j} \mathbf{x}_{\mathbf{i};j} B_j^{m_x}(v) B_{\mathbf{i}}^{n_x}(\mathbf{u})}{\sum_{j=0}^{m_x} \sum_{|\mathbf{i}|=n_x} w_{x\mathbf{i};j} B_j^{m_x}(v) B_{\mathbf{i}}^{n_x}(\mathbf{u})} \text{ and} \\ \mathbf{m}(\mathbf{u}; v) &= \frac{\sum_{j=0}^{m_m} \sum_{|\mathbf{i}|=n_m} w_{m\mathbf{i};j} \mathbf{x}_{\mathbf{i};j} B_j^{m_m}(v) B_{\mathbf{i}}^{n_m}(\mathbf{u})}{\sum_{j=0}^{n_m} \sum_{|\mathbf{i}|=n_m} w_{m\mathbf{i};j} B_j^{m_m}(v) B_{\mathbf{i}}^{n_m}(\mathbf{u})}. \end{aligned} \tag{4.33}$$

The coordinate $\mathbf{u}$ is a parametric point within the cross-section of the wedge and the coordinate v is the parametric distance along the length of the wedge. Regions defined by this class can considered as sweeps or lofting of a Bézier triangles along Bézier curves (see the class definitions above). The degree of the shape (n_x) or composition (n_m) variation of over the cross-section is independent from the vacation along the length of the wedge (m_x or m_m).

The number of control points and weights required to define a Bézier pentahedron is simply the product of the number of control points to define a cross section (see Equation 4.24) and the order of the variation along its length:

$$\sum_{j=0}^{m} \sum_{|\mathbf{i}|=n} 1 = \frac{1}{2}(m+1)(n+1)(n+2). \tag{4.34}$$

Knowing the number of control points and weights as a function of degree, the storage cost for

representing an FGMRationalBézierPent region is determined to be:

$$S_{rbpr} = 4S_{int} + \left[\frac{4(m_x+1)(n_x+1)(n_x+2) + (d_m+1)(m_m+1)(n_m+1)(n_m+2)}{2}\right] S_{flt}. \quad (4.35)$$

FGM Rational Bézier Hexahedron: The final FGMDomain based on a Bézier formulation if the FGMRationalBézierHex. Objects of this class define the shape and composition of three dimensional topological objects homeomorphic to a cube (a brick). The shape and composition of the region are defined by lattices of control points and weights in Build Space $(\mathbf{x}, \mathbf{w}_x)$ and Material Space $(\mathbf{m}, \mathbf{w}_m)$. The mapping is an extension of that for the quadrilateral path (Equation 4.26 into a higher dimension; from a unit cube into a region in Build Space according to:

$$\begin{aligned} \mathbf{x}(\mathbf{u}) &= \frac{\sum_{i_0=0}^{l_x}\sum_{i_1=0}^{m_x}\sum_{i_2=0}^{n_x} w_{x i_0 i_1 i_2}\mathbf{x}_{i_0 i_1 i_2} B_{i_0}^{l_x}(u_0) B_{i_1}^{m_x}(u_1) B_{i_2}^{n_x}(u_2)}{\sum_{i_0=0}^{l_x}\sum_{i_1=0}^{m_x}\sum_{i_2=0}^{n_x} w_{x i_0 i_1 i_2} B_{i_0}^{l_x}(u_0) B_{i_1}^{m_x}(u_1) B_{i_2}^{n_x}(u_2)} \text{ and} \\ \mathbf{m}(\mathbf{u}) &= \frac{\sum_{i_0=0}^{l_m}\sum_{i_1=0}^{m_m}\sum_{i_2=0}^{n_m} w_{m i_0 i_1 i_2}\mathbf{m}_{i_0 i_1 i_2} B_{i_0}^{l_m}(u_0) B_{i_1}^{m_m}(u_1) B_{i_2}^{n_m}(u_2)}{\sum_{i_0=0}^{l_m}\sum_{i_1=0}^{m_m}\sum_{i_2=0}^{n_m} w_{m i_0 i_1 i_2} B_{i_0}^{l_m}(u_0) B_{i_1}^{m_m}(u_1) B_{i_2}^{n_m}(u_2)}. \end{aligned} \quad (4.36)$$

The coordinate $\mathbf{u} = (u_0, u_1, u_2)$ is the parametric location of a point within a unit cube the Cartesian space. The index of the control point is represented by $\mathbf{i} = (i_0, i_1, i_2)$ The degrees of variation in shape and composition along the $\hat{\mathbf{u}}_0$, $\hat{\mathbf{u}}_1$, and $\hat{\mathbf{u}}_2$ are l_x,m_x, and n_x; and l_m,m_m, and n_m, respectively.

The total storage cost for an FGMRationalBézierHex region is:

$$S_{rbhr} = 6S_{int} + [4(l_x+1)(m_x+1)(n_x+1) + (d_m+1)(l_m+1)(m_m+1)(n_m+1)]\, S_{flt}. \quad (4.37)$$

General 3D Cell or B-rep Region: To efficiently represent regions of uniform composition, a single composition vector can be used and the shape of the region inferred from the topological data structure containing the lower dimensional entities (its boundary). This is analogous to how composite structures are currently represented within B-rep modeling systems. Within the framework described here, the FGMBRepRegion domain is introduced. This FGMDomain simply defines a vector of material volume fractions $(\mathbf{m})$ for the corresponding Region or Cell (r) stored in the topological data structure reference by the domain. The composition of the region, interior to the bounding Cells (or Shells) stored within the generalized solid modeling data structure, is considered uniform or piece-wise constant. In this way, large regions of uniform composition can be defined without further subdivision of the model into simpler FGMDomains. The storage cost of each such region is:

$$S_{cr} = S_{ptr} + d_m S_{flt}. \quad (4.38)$$

Summary of FGMDomain storage costs

The storage cost of each derived class of FGMDomain depends on the amount of data each class maintains as well as the complexity (degree) of the composition and shape variation. These storage costs are summarized in Table 4.3.

FGMDomain	Storage cost
FGMPoint	$(3 + d_m)S_{flt}$
FGMRationalBézierCrv	$2S_{int} + [4(n_x + 1) + (d_m + 1)(n_m + 1)]S_{flt}$
FGMRationalBézierTri	$2S_{int} + \left[2(n_x + 1)(n_x + 2) + \frac{d_m+1}{2}(n_m + 1)(n_m + 2)\right] S_{flt}$
FGMRationalBézierQuad	$4S_{int} + [4(m_x + 1)(n_x + 1) + (d_m + 1)(m_m + 1)(n_m + 1)] S_{flt}$
FGMPlanarSurface	$S_{ptr} + (d_m + 4)S_{flt}$
FGMRationalBézierTet	$2S_{int} + \left[\frac{4(n_x+1)(n_x+2)(n_x+3)+(d_m+1)(n_m+1)(n_m+2)(n_m+3)}{6}\right] S_{flt}$
FGMRationalBézierPent	$4S_{int} + \left[\frac{4(m_x+1)(n_x+1)(n_x+2)+(d_m+1)(m_m+1)(n_m+1)(n_m+2)}{2}\right] S_{flt}$
FGMRationalBézierHex	$6S_{int} + [4(l_x + 1)(m_x + 1)(n_x + 1)+ (d_m + 1)(l_m + 1)(m_m + 1)(n_m + 1)] S_{flt}$
FGMBRepRegion	$S_{ptr} + d_m S_{flt}$

Table 4.3: Storage costs of FGMDomain definitions used in analysis of memory requirements for generalized FGM modeling methods.

Other definitions for FGMDomains

Only a small subset of all the possible FGMDomains that could be defined have been presented here. These FGMDomains were based on rational Berstein polynomials were chosen for their flexibility in representing a wide range of shapes accurately (cylindrical and spherical patches [15], for instance). For some applications, the rational formulation may not be needed. In addition, material information associated with lower dimensional entities (vertices, curves, and faces) may not be needed but is included for consistency. Additional FGMDomains could also be defined to further extend the generalized cellular approaches to solid modeling, just as the STEP standard includes many representations for sphape (see Figure 2-4). Other parameteric FGMDomains could be based on NURBS (Non-Uniform Rational B-Spline) [15, 16, 19, 52, 55, 69] or simplex spline [14, 61] representations. The Bernstein FGMDomains are special cases of these two. Procedural methods, similar to the offset surfaces [45, 51] used in exisiting solid model representations [49], could also be introduced.

4.6.3 Relationship between FGMDomains and topology

At the beginning of this section, the generality of the Radial-Edge and Cell-Tuple-Graph data structures was emphasized, permitting a decoupling between the representation of the topology of the model and the shape and composition. The definition of the shape and composition in an FGMDomain, however, must map completely onto the corresponding entity in the topological data structure, defining the shape and composition to the entities boundary. In order to guarantee that

this happens, the topology of the FGMDomain must be homeomorphic to the corresponding entity in the relational database. Therefore, since the parametric FGMDomains discussed in this section all map a parametric space of a given topology into the topological data base, the number of objects stored in the database can be related to each instance of an FGMDomain in the model. This section lists the number of topological entities generated for each case of an FGMDomain.

Relationship between FGMDomains and RadialEdge objects

The Radial-Edge data structure maintains the two major sets of classes: topological entities and their uses. The number instances of each entity and use created for each instance of an FGMDomain are listed in Tables 4.4 and 4.5. The relationship between the parametric FGMDomains and topological objects is fixed. Each instance of a point, curve, parametric surface, or parametric regions results in a since instance of a vertex, edge, face, or region, respectively. Each surface or region also requires an object to represent its boundaries; either a Loop or Shell.

As for the roles of each topological entity, the number of Loopuses, Edgeuses, and Vertexuses depend on the topologies of the surfaces and the number of Faceuses depend on the topologies of regions. For a two dimensional FGMDomain (a surface), each Edge in a Face contributes 2 Edgeuses and 2 Vertexuses to the relational database. In addition, two Loopuses are also generated to define the opposing orientations for each Loop bounding the Face. Therefore, each FGMDomain corresponding to a Face contributes twice as many Edgeuses and Vertexuses as it has bounding Edges, and twice as many Loopuses as it has Loops. Similarly, each three dimensional FGMDomain contributes one Faceuse for each of its Faces, defining the relative orientation of that Face.

For the implicit FGMDomains that derive their topology from the topological database, however, the relationship is entirely dependent upon how the FGM object is decomposed in the FGMDomain. An FGMBRepRegion, for instance, may be bounded by an arbitrary number of Faces and possibly by multiple Shells (in order to represent voids within the region). Likewise, an FGMPlanarSurface may be bounded by an arbitrary number of Edges and may also contain holes (defined by additional Loops of Edges forming internal boundaries).

Relationship between FGMDomains and Cell-Tuple-Graph objects

The Cell-Tuple-Graph data structure maintains the topology of the model in graphs of Tuples, each of which represent a unique combination of incident Cells in the model of different dimensions. The relationship between the objects in this data structure and each class of FGMDomain is listed in Table 4.6. Each FGMDomain is represented by a Cell in the Cell-Tuple-Graph data structure, resulting in a one-to-one correspondence between the number of Cells and FGMDomains. The Tuples, or unique incident states, is related to number of two dimensional Cells (Faces) in the model. For each Edge bounding a Face, 4 Tuples are required to represent the four incidence states,

FGMDomain	$\frac{Region}{instance}$	$\frac{Shell}{instance}$	$\frac{Face}{instance}$	$\frac{Loop}{instance}$	$\frac{Edge}{instance}$	$\frac{Vertex}{instance}$
FGMPoint						1
FGMRationalBézierCrv					1	
FGMRationalBézierTri			1	1		
FGMRationalBézierQuad			1	1		
FGMPlanarSurface			1	$n_{holes}+1$		
FGMRationalBézierTet	1	1				
FGMRationalBézierPent	1	1				
FGMRationalBézierHex	1	1				
FGMBRepRegion	1	$n_{voids}+1$				

Table 4.4: The relationships between the number of topological entities in Radial-Edge data structure and each derived class of FGMDomain.

FGMDomain	$\frac{Faceuse}{instance}$	$\frac{Loopuse}{instance}$	$\frac{Edgeuse}{instance}$	$\frac{Vertexuse}{instance}$
FGMPoint				
FGMRationalBézierCrv				
FGMRationalBézierTri	2	2	6	6
FGMRationalBézierQuad	2	2	8	8
FGMPlanarSurface	2	$2(1+n_{holes})$	$2n_{edges}$	$2n_{edges}$
FGMRationalBézierTet				
FGMRationalBézierPent				
FGMRationalBézierHex				
FGMBRepRegion				

Table 4.5: The relationships between the number roles of each topological entities in Radial-Edge data structure and each derived class of FGMDomain.

two at either end of the Edge and two on either side of the Face. The number of isolated graphs of Tuples (CTGs) in the model depends upon its topology. If all of the incident states are connected (no voids in Regions or holes in Faces), then only one CTG exists. Such a model is considered to be a subdivided manifold. Finite element meshes are one subset of subdivided manifold.

With the introduction of implicit FGMDomains (FGMPlanarSurface and FGMBRepRegion), the possibility for disconnected graphs exists. The incident states for the topology for a void, for instance, are clearly isolated from the Tuples in the Regions exterior boundary. The relationship between CTG's and FGMPlanarSurfaces, however, is not fixed (indicated by the "X" in the Table). Paths of connected Tuples between a Face's exterior and interior loops may exist depending on how the surfaces bounding the Face's hole through the incident region connect to the boundary of the region. Therefore, no definite relationship can be asserted, depending instead on how the object is decomposed into Cells.

4.7 Discussion

The preceding sections develop expressions for the memory requirements for several different modeling schemes, beginning with a simple exhaustive enumeration approach to general B-rep and cellular

FGMDomain	$\frac{ctg's}{instance}$	$\frac{cells}{instance}$	$\frac{tuples}{instance}$
FGMPoint		1	
FGMRationalBézierCrv		1	
FGMRationalBézierTri		1	12
FGMRationalBézierQuad		1	16
FGMPlanarSurface	X	1	$4n_{edges}$
FGMRationalBézierTet		1	
FGMRationalBézierPent		1	
FGMRationalBézierHex		1	
FGMBRepRegion	$n_{voids}+1$	1	

Table 4.6: The relationships between the number instances of each class in the Cell-Tuple-Graph data structure and each derived class of FGMDomain.

decomposition methods. The memory required for each of the representations, in terms of the number of instances of their fundamental data classes, is summarized in Table 4.7. Note that the generalized data structure storage requirements (Radial-Edge and Cell-Tuple-Graph) only represent the cost associated with maintaining the topology explicitly; the representation of the shape and composition are depends entirely on the nature of the model and how it was constructed. The term $\sum_{fgmd \in model} S_{fgmd}$ represents the total cost associated with all of the FGMDomains used to model the object. In addition, the definition of the FGMDomains also associate material information with entities of zero, one, and two dimensions. This information is included for consistency of FGMDomain definitions (all FGMDomains maintain shape and composition information), the unabmiguous definition of the composition at each point in the Build Space (each point in BuildSpace maps uniquely to one Cell and its associated FGMDomain), and to follow the paradigm set forth in B-rep modeling in which analytic descriptions of vertex and curve geometries are explicitly maintained even though this information is implicitly conveyed with the intersection of the higher order surfaces. This generality allows the greatest freedom for the definition of future FGMDomain representations and design methods which may employ material information associated with lower dimensional entities.

The four finite element methods presented in this chapter are slight variation of each other. Two methods incorporate backwards pointers from the elements to the vertices. This information may be helpful in the optimal design of algorithms for working with the data structure (this is beyond the scope of this thesis). The other variation deals with where the material information is maintained. In one pair of of finite element data structures, the information is associated with the vertices, similar to how field values are associated with nodes in a finite element mesh. The composition of each element is determined from blends of the values at the noes. In the other pair, the material information is directly assocatied with each tetrahedron. How the material information is associated with the elements greatly impacts how the mesh can represent discontinuities in material composition, as shown in Figure 4-20.

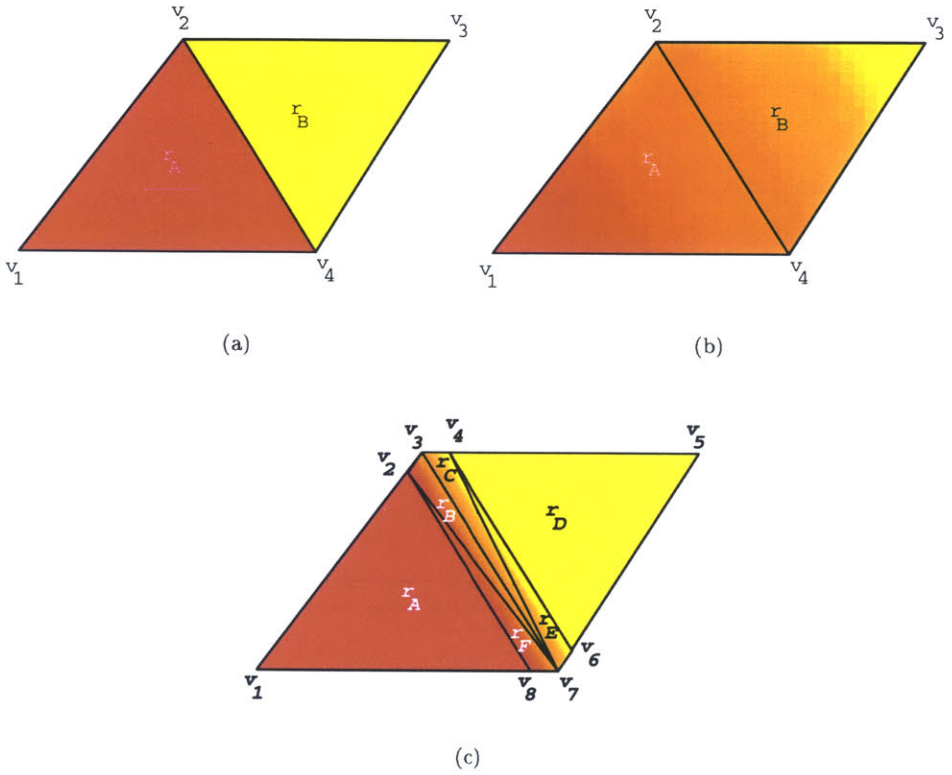


Figure 4-20: (a) Modeling two piecewise constant regions with composition information associated with (a) the two regions and (b) the vertices. In order to represent two piece-wise constant regions when the material information is associated with the vertices, the interface region must be meshed as in (c).

Exhaustive Enumeration:
$S_{vox} = 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{1}{8} n_{vox} d_m \lceil \lg n_\lambda \rceil$
Triangulated B-rep Model:
$S_{tri} = (S_{int} + S_{ptr} + d_m S_{flt}) n_r + 8 S_{ptr} n_{triangles} + 3 S_{flt} n_{nodes} + S_{int} + S_{ms}$
FE-Model:
$S_{te_{00}} = 4 S_{ptr} n_{tetrahedra} + (3 + d_m) S_{flt} n_{nodes} + S_{int} + S_{ms} + S_{ptr}$ (FE-TET, FE-Vert-M)
$S_{te_{01}} = 4(S_{ptr} + d_m S_{flt}) n_{tetrahedra} + 3 S_{flt} n_{nodes} + S_{int} + S_{ms} + S_{ptr}$ (FE-TET-M, FE-Vert)
$S_{te_{10}} = 8 S_{ptr} n_{tetrahedra} + [(3 + d_m) S_{flt} + S_{int}] n_{nodes} + S_{int} + S_{ms} + S_{ptr}$ (FE-TET, FE-Vert-MP)
$S_{te_{11}} = 4(2 S_{ptr} + d_m S_{flt}) n_{tetrahedra} + [3 S_{flt} + S_{int}] n_{nodes} + S_{int} + S_{ms} + S_{ptr}$ (FE-TET-M, FE-Vert-P)
Radial-Edge:
$S_{re} = S_{ptr} + S_{ms} + 5 S_{ptr} n_r + 4 S_{ptr} n_s + 14 S_{ptr} n_f + S_{ptr} n_l + 2 S_{ptr} n_e + 2 S_{ptr} n_v + 6 S_{ptr} n_{lu} + 7 S_{ptr} n_{eu} + 4 S_{ptr} n_{vu} + S_{ms} + \sum_{fgmd \in model} S_{fgmd}$
Cell-Tuple-Graph:
$S_{ctg} = 2 S_{ptr} n_{ctg} + \sum_{t \in model} [S_{int} + (2t.d + 3) S_{ptr}] + \sum_{c \in model} [2 S_{int} + (c.n_g + 1) S_{ptr}] + S_{ms} + \sum_{fgmd \in model} S_{fgmd}$

Table 4.7: Memory requirements for Exhaustive Enumeration, Triangulated B-Rep, Tetrehedral Mesh, Radial-Edge, and Cell-Tuple-Graph solid modeling methods.

Chapter 5

Bounds for voxel-based model growth

5.1 Motivation

Exhaustive enumeration modeling methods represent objects as a lattice of voxels. Filled voxels define interior sub-regions of the model while empty voxels are outside the model. In most applications of voxelized representations, the dimensions of each voxel and the total number of voxels are determined by the resolution of some inspection process (eg. medical imaging) [37]. For design applications, voxel dimensions remain variables to be determined by the designer. Since the storage cost of an exhaustive enumeration method is related to the voxel size, the factors influencing the selection of voxel size are explored to relate memory requirements to parameters more directly related to the quality of the modeled object representation.

5.2 Voxel size dictates lattice size

In order to represent an object as a voxelized model, the object must be placed within a three dimensional lattice of voxels. Each voxel within the lattice defines the compositions of the corresponding sub-region within the FGM object. The total number of voxels within the lattice (n_{vox}) is a functions of the physical size of the lattice ($L_x \times L_y \times L_z$) and the size of each voxel ($\delta_x \times \delta_y \times \delta_z$). The physical size of the lattice is determined by the bounding box of the object to be modeled. Given that the n_{vox} voxels must fill the space, the following inequality must hold:

$$n_{vox}\delta_x\delta_y\delta_z \geq L_xL_yL_z. \tag{5.1}$$

Therefore, the total number of voxels (one factor affecting the storage cost) is inversely related to the dimensions of the voxels.

$$n_{vox} = \left\lceil \frac{L_x}{\delta_x} \right\rceil \left\lceil \frac{L_y}{\delta_y} \right\rceil \left\lceil \frac{L_z}{\delta_z} \right\rceil \tag{5.2}$$

5.3 Geometric constraint on voxel size

Voxel-based modeling provides a discrete representation of an object. Before investigating this discretization's impact on the representation of composition, its impact on single material modeling must be understood. In a model consisting of a single material, the nature of the material boundary defining the part surface is the most important piece of conveyed information. As a model is designed (when the bounding surface is defined and modified), voxels are set as filled or empty depending upon where they lie relative the the desired boundary. Obviously, all voxels that are completely interior to the boundary are considered filled and do not present a problem. Voxels on the boundary, however, may or may not be filled depending upon how they intersect the desired, designed boundary. To understand which voxels get filled, let us define an algorithm that fills voxels to represent a new feature added to the model, as described in Algorithm 1. The criteria used to determined whether or not to change the state of a voxel to "filled" depends upon how much of a given voxel lies interior to the new feature's boundary, as illustrated in Figure 5-1.

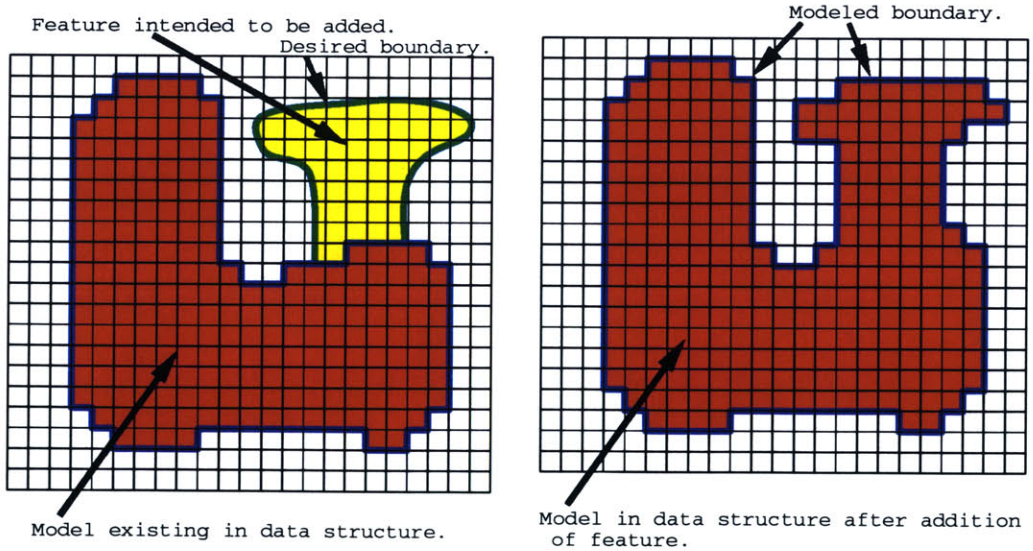


Figure 5-1: (a) The addition of a feature to a voxel-based data structure. (b) Modified voxel-based model to capture intended feature.

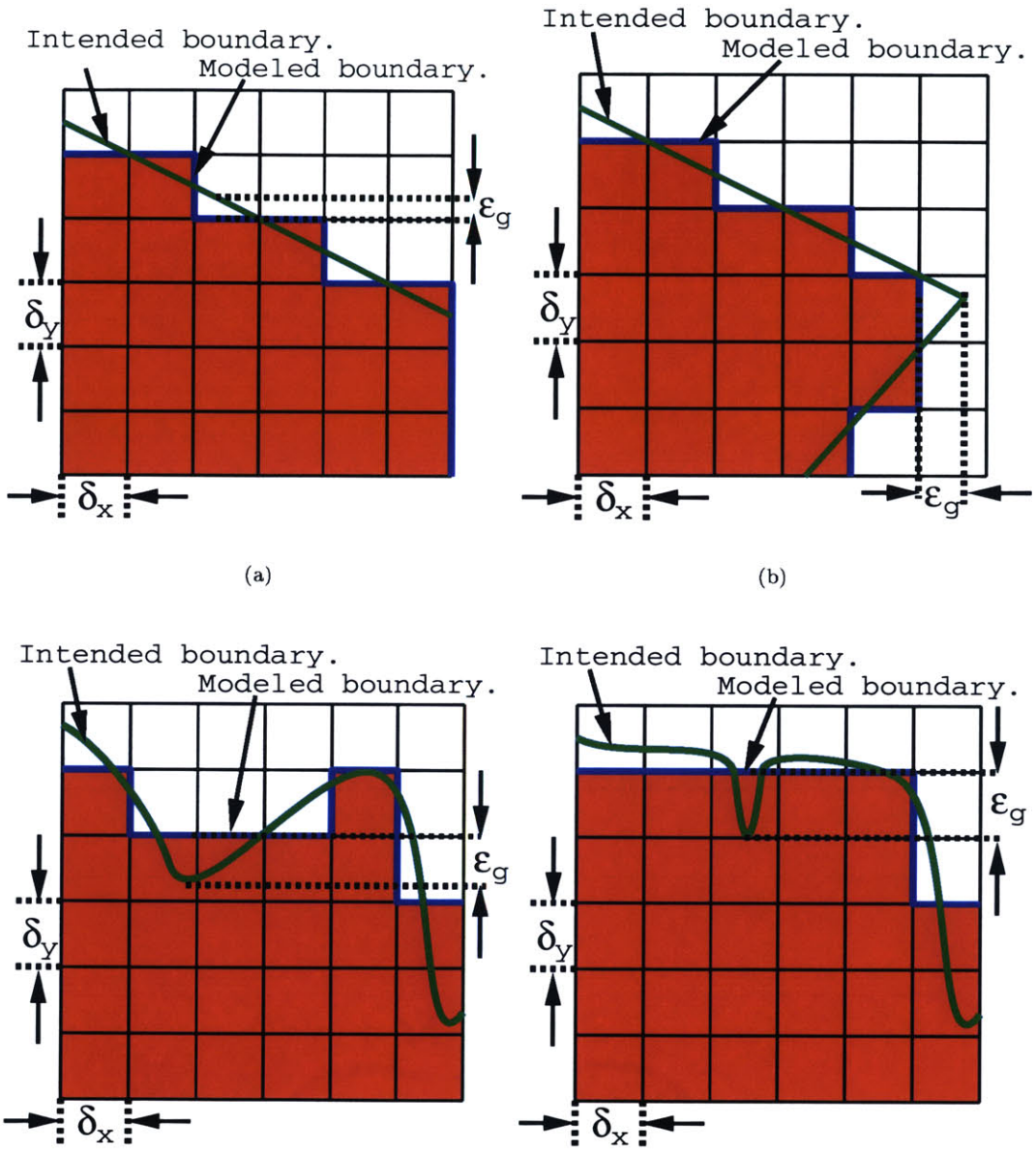


Figure 5-2: Examples of distretization of the intended boundaries of models into voxels.

Algorithm 1 addFeature(**lattice**, *feature*); add a single material "feature" to a binary voxel model.

Require: **lattice** is a 3D lattice of binary voxels. Each voxel represents the presence or not of material: *filled* or *empty*.
feature references some feature the designer wishes to be added to the model.
Ensure: The voxels in **lattice** that are determined to be interior to the feature are set to *filled* in order to represent the feature.

```
1: for each v_ijk ∈ lattice do {Iterate through the voxels.}
2:   if (v_ijk ∩ feature) > ½(δ_x δ_y δ_z) then {Is majority of voxel within feature?}
3:     v_ijk ← filled {Yes, set the current voxel as filled.}
4:   else {No, do not modify voxel.}
5:     v_ijk ← v_ijk
```

If the feature only partially fills a voxel, the discretized boundary over the filled or empty voxel will be in error, some distance ϵ_g from the desired boundary of the feature. This error is a function of the shape of the feature's surface through the voxel, but will at most be the length of the maximum dimension of the voxel, as shown in Figure 5-2 for several discretization cases. The overall accuracy of the model is determined by the maximum deviation of the boundary from the intended to the modeled. Since the walls of the voxels are parallel to the axes, this distance is always measured parallel to the axes. Therefore, the accuracy at which a voxelized model represents the designer's intent is a function of the resolution of the lattice (size of each voxel) and is limited by the worst case (Figure 5-2), yielding the following relationship between the voxel dimensions and the geometric accuracy:

$$\epsilon_g = \max\{\delta_x, \delta_y, \delta_z\}. \tag{5.3}$$

Rewriting the above equation, we can express the maximum dimensions for the voxels as a function of a desired accuracy for representing the geometry (shape) of a model.

$$\delta_x \leq \epsilon_g,\ \delta_y \leq \epsilon_g,\ \delta_z \leq \epsilon_g \tag{5.4}$$

Therefore, the geometric accuracy imposes an upper limit on the dimension of the voxels.

In addition, to capture the geometry of small features in the model, the voxel dimension must be less than or equal to the minimum feature size in the model (see Figure 5-3). Therefore, the dimensions of the voxels must also satisfy another constraint:

$$\delta_x \leq \mu_g,\ \delta_y \leq \mu_g,\ \delta_z \leq \mu_g \tag{5.5}$$

where μ_g is the minimum geometric feature size in the intended design.

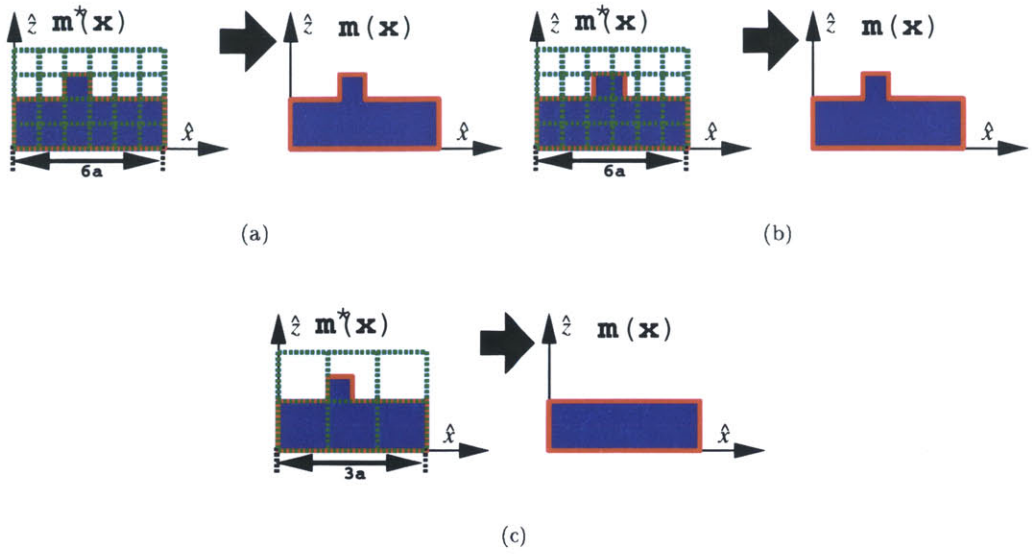


Figure 5-3: Intended designs for object boundary ($\mathbf{m}^*(\mathbf{x})$) and modeled boundary ($\mathbf{m}(\mathbf{x})$). The dimensions of the voxels are $\delta_x, \delta_y, \delta_z = a$. (a.) Boundary of geometric feature lies along voxel boundaries. (b.) Boundary of geometric feature lies off voxel boundaries but is still captured in representation. (c.) Voxel mesh is too coarse to capture geometric feature.

5.4 Composition constraint on volume fraction resolution

To model FGM objects, vectors of volume fractions of the base materials in the material system are assigned to each voxel, permitting the representation of *compositions* within a *grey-valued* voxel model. The resolution in composition, however, is limited to the number of levels of intensity allocated to each voxel. Even though the designer may wish to specify any volume fraction for each material, the discrete representation of intensity levels results in the continuous value being thresholded to the closest level representable by the voxelized system. This thresholding effect is illustrated in Figure 5-4.

To quantify this error, let us assume that the desired presence (volume fraction) of a single material (m_l^*) can take on any value between zero and one: $0.0 \leq m_l^* \leq 1.0$. A voxel, however, can only represent n_λ discrete volume fractions, requiring that the desired volume fraction be thresholded such that the voxel's assigned value is:

$$m_l = \frac{\lfloor m_l^*(n_\lambda - 1) + \frac{1}{2} \rfloor}{n_\lambda - 1} \tag{5.6}$$

$$\text{where} \quad m_l^* \in [0,1]$$

$$\text{and } m_l \in \{0.0, \frac{1}{n_\lambda - 1}, \frac{2}{n_\lambda - 1}, \ldots 1.0\}$$

Although a designed composition (m_l^*) may be exactly representable as one of the discrete levels of the voxels, it is more likely that there will be some difference between the desired composition and

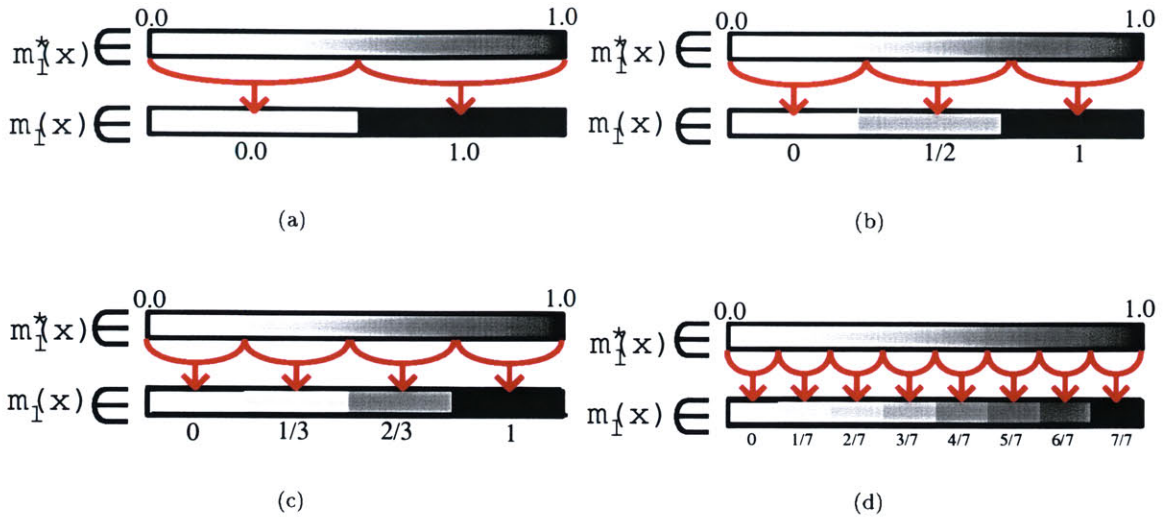


Figure 5-4: Thresholding of continuous grading to discrete levels maintained in voxel representation (a) $n_\lambda = 2$, (b) (a) $n_\lambda = 3$, (b) (a) $n_\lambda = 4$, (d) $n_\lambda = 7$.

a representable value: $m_l^* \neq m_l$. This difference leads to an error in the material volume fraction assignment and is inversely proportional to the number of levels of intensity that can be represented:

$$|m_l^* - m_l| \leq \epsilon_m = \frac{1}{2(n_\lambda - 1)} \tag{5.7}$$

In order to guarantee a certain accuracy in composition representation, therefore, a minimum resolution of material volume fractions must be supported, as given by:

$$n_\lambda = \left\lceil \frac{1}{2\epsilon_m} + 1 \right\rceil \tag{5.8}$$

Algorithm 2 Add a multiple material "feature" to an FGM voxel model: addFeature(**lattice**, *feature*)

Require: **lattice** is a 3D lattice of voxels. Each voxel represents the d_m volume fractions corresponding to materials in the material system, each at one of n_λ intensity levels.
feature is some feature the designer wishes to add to the model. A vector value function ($\mathbf{m}^*(\mathbf{x})$) associated with the feature provides a mapping from Build Space into Material Space.

Ensure: The voxels in **lattice** that are determined to be interior to the feature are set to a thresholded composition which best approximates the feature's composition.

1: **for** each $v_{ijk} \in$ **lattice** **do** {Iterate through the voxels.}
2: **if** $(v_{ijk} \cap feature) > \frac{1}{2}(\delta_x \delta_y \delta_z)$ **then** {Is majority of voxel within feature?}
3: **for** $l \leftarrow 0$ to $d_m - 1$ **do** {Yes, loop through the materials in system.}
4: $\nu_l \leftarrow \iiint_{voxel} feature.m_l dV/(\delta_x \delta_y \delta_z)$ {Compute presence of m_l in voxel.}
5: $v_{ijk}.m_l \leftarrow \frac{\lfloor \nu_l (n_\lambda - 1) + \frac{1}{2} \rfloor}{n_\lambda - 1}$ {Threshold the desired volume fraction and assign.}
6: **else** {No, do not modify voxel.}
7: $v_{ijk} \leftarrow v_{ijk}$

5.5 Composition constraint on voxel size

A desired accuracy in representing the composition also affects the size of the voxels used to approximate the material variation throughout the model. Obviously the smaller the voxels, the closer the grading will be to the ideal, desired grading. The relationship between voxel size and composition accuracy can be quantified and depends on the nature of the grading. In this section, we explore the impact of the desired grading on the required size of the voxels in order to guarantee a certain degree of composition accuracy.

5.5.1 Constraint based on discontinuities in composition

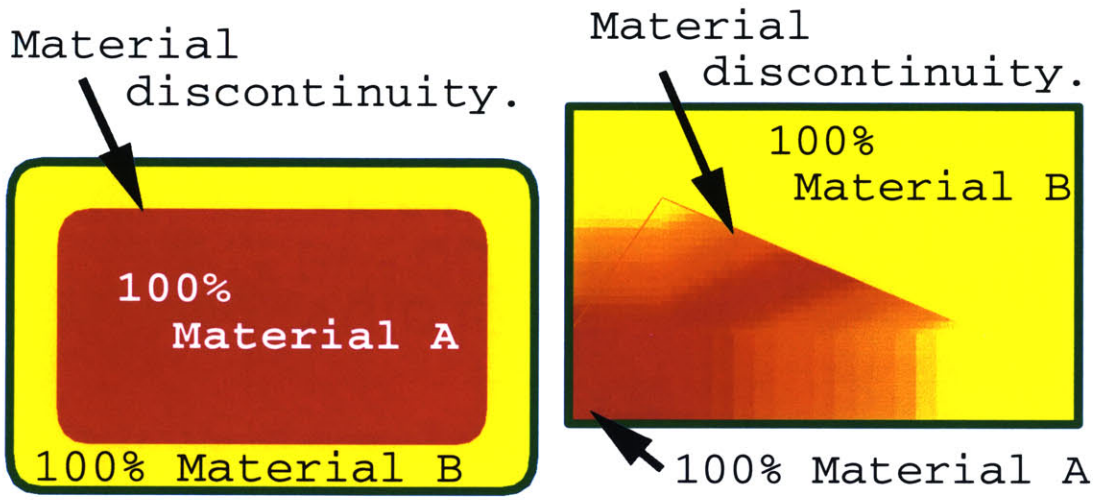
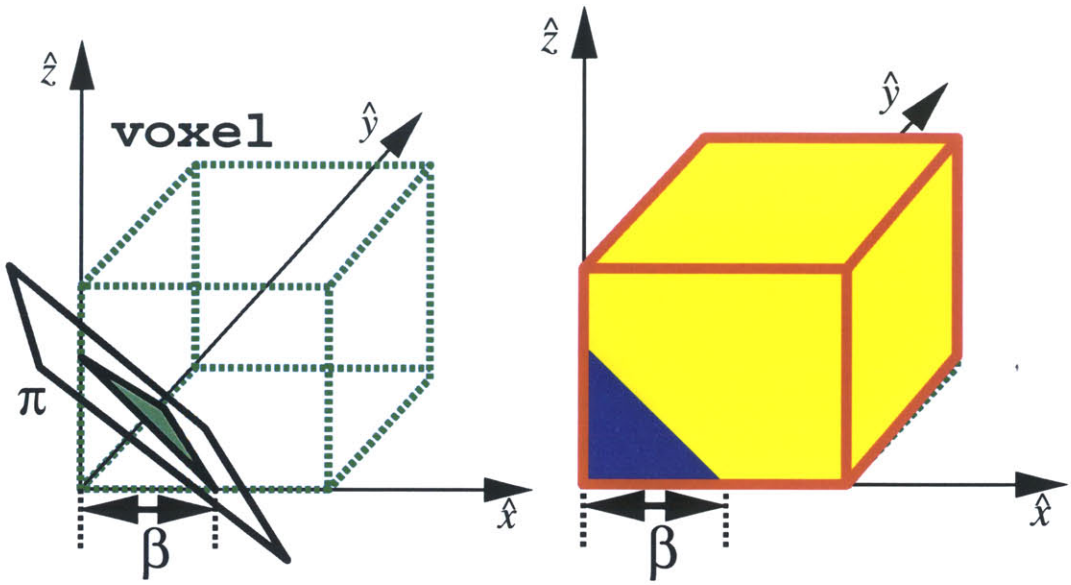


Figure 5-5: Examples of discontinuities in composition. (a) Discontinuity between two regions of uniform composition. (b) Discontinuity with regions of graded composition.

The first constraint on composition stems from discontinuities in composition (points in the model where $\vec{\nabla}\mathbf{m}(\mathbf{x})$ is not defined) as illustrated in Figure 5-5. This constraint is analogous to the geometric constraint described above for a single material, except in this case the boundary is internal, separating two regions of differing composition rather than defining the external surface of the model. Referring to Lines 4-5 of Algorithm 2, we can formulate an expression for the voxel size in terms of a material accuracy: ϵ_m.

First, let us consider a voxel enclosing the subregion $x_i \leq x < x_i + a$, $y_j \leq y < y_j + a$, $z_k \leq z < z_k + a$, as shown in Figure 5-6(a). We will assume the voxel lies wholly within a feature of graded composition which is being designed. Let us define a hypothetical composition variation for this feature within the voxel such that the desired variation has an internal boundary between regions



(a) (b)

Figure 5-6: (a) Voxel to approximate a subregion of discontinuous composition. The discontinuity in composition occurs of the plane π. (b) The desired composition ($\mathbf{m}^*(\mathbf{x})$) over the subregion occupied by the voxel.

of differing composition:

$$\mathbf{m}^*(x,y,z) = \begin{cases} \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} & \text{if } x - x_i + y - y_j + z - z_k < \beta a \\ \begin{bmatrix} 0.0 \\ 1.0 \end{bmatrix} & \text{otherwise} \end{cases} \qquad \text{such that } 0.0 \leq \beta \leq 1.0 \qquad (5.9)$$

Figures 5-6(b) shows this desired material distribution for the voxel. Since a voxel can represent only a single composition [1], Algorithm 2 uses the average composition of the intended design to assign to the voxel.

$$\nu_l = \frac{\int_{x_i}^{x_i+a}\int_{y_j}^{y_j+a}\int_{z_k}^{z_k+a} m_l^*(x,y,z)dxdydz}{a^3}$$

$$\begin{bmatrix} \nu_0 \\ \nu_1 \end{bmatrix} = \begin{bmatrix} \frac{\beta^3}{6} \\ 1.0 - \frac{\beta^3}{6} \end{bmatrix}$$

In addition, since a voxel can only represent discrete values, this average composition must be

[1] The composition $\mathbf{m}$ associated with voxel v_{ijk} is expessed as $v_{ijk}.\mathbf{m}$.

thresholded:

$$v_{ijk}.\mathbf{m} = \begin{bmatrix} \frac{\lfloor \frac{\beta^3}{6}(n_\lambda - 1) + \frac{1}{2} \rfloor}{n_\lambda - 1} \\ \frac{n_\lambda - \lfloor \frac{\beta^3}{6}(n_\lambda - 1) + \frac{1}{2} \rfloor - 1}{n_\lambda - 1} \end{bmatrix} \tag{5.10}$$

Through studying this composition assignment process, we discover two artifacts due to the voxelized representation of the composition discontinuity: loss of accuracy in composition and loss of information about the interface boundary between discontinuous regions. Although both involve material assignment, the latter can be attributed to the minimum material feature size.

The first error is due to a difference between the assigned and the intended compositions:

$$\epsilon_m = \max\{m_0 - m_0^*(x,y,z), m_1 - m_1^*(x,y,z) | (x,y,z) \in ([x_i, x_i + a), [x_i, x_i + a), [x_i, x_i + a))\} \tag{5.11}$$

Figures 5-7(a)-(c) shows the assigned volume fractions for different values of β. When $\beta \leq \sqrt[3]{\frac{3}{n_\lambda - 1}}$,

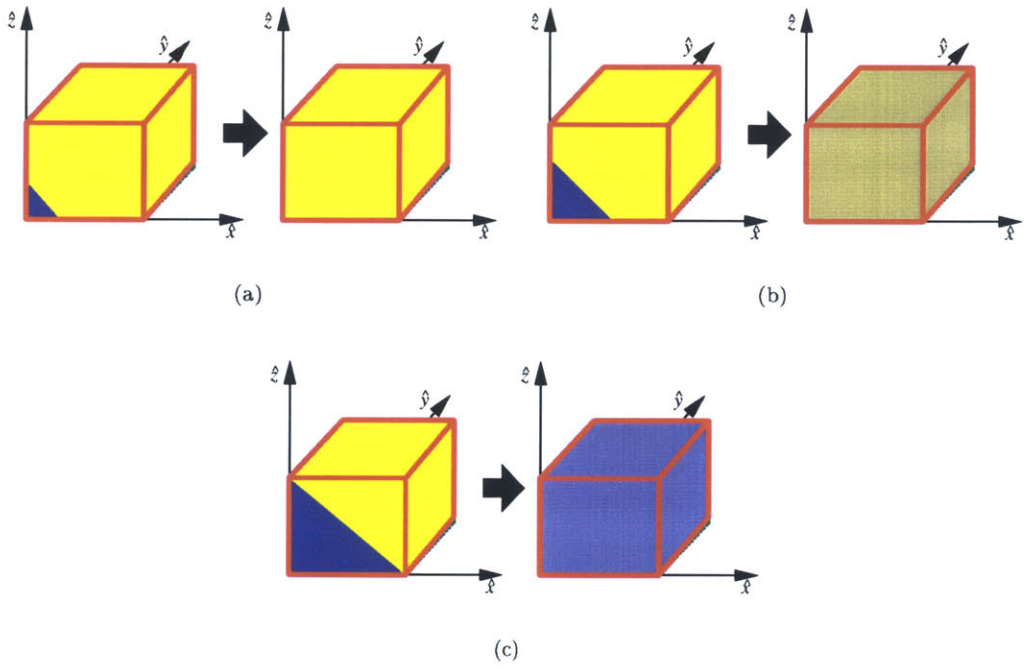


Figure 5-7: Thresholding of designed material assigned to discretized regions of uniform composition for (a) $\beta \leq \sqrt[3]{\frac{3}{n_\lambda - 1}}$, (b) $\beta = \frac{1}{2}$, and (c) $\beta = 1$.

we find that the material error is at its greatest, $\epsilon_m = 1$, over the region $x - x_i + y - y_j + z - z_k \leq \beta a$. Since β is a free parameter, subject only to the designer's intent ($0.0 \leq \beta \leq 1.0$), no voxel size, no matter how small, can guarantee an improvement in composition accuracy. For this reason, a material accuracy cannot be considered as a valid constraint for restricting voxel size within FGM

models containing *discontinuous* compositions.

The second effect of discretization concerns the material feature represented by the internal boundary. As explained in the previous section, a voxel can only represent boundaries along a combination of its eight faces. A design that includes a discontinuity in composition across a surface has this surface as an internal boundary between regions of differing composition and can be considered, itself, a geometric feature. For the example given above (Equation 5.9), this design feature is the section of the plane ($\pi : x - x_i + y - y_j + z - z_k = \beta a | x_i \leq x < x_i + a,\ y_j \leq y < y_j + a,\ z_k \leq z < z_k + a$). In the voxelized representation, however, this feature consists of three of the faces of the voxel. Even if the composition relative to either side of this feature can be represented exactly, the shape of this feature becomes perturbed from the intended design. In this planar case, the largest distance this feature will be perturbed is one half the dimension of the voxel. For more general cases (for the design of curved or faceted interfacial surfaces, for instance) this error grows to that observed before at the object's external boundary (Equation 5.3). Therefore, to accurately represent the interface at discontinuities in composition, the size of the voxels in the model is limitied by the intended minimum material feature size in the model:

$$\delta_x \leq \mu_m,\ \delta_y \leq \mu_m, \text{ and } \delta_z \leq \mu_m. \tag{5.12}$$

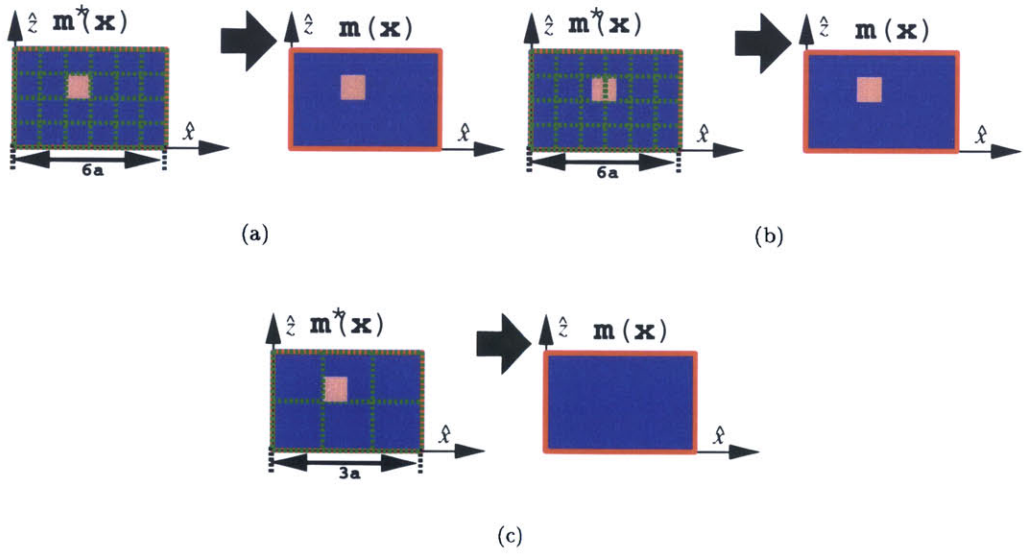
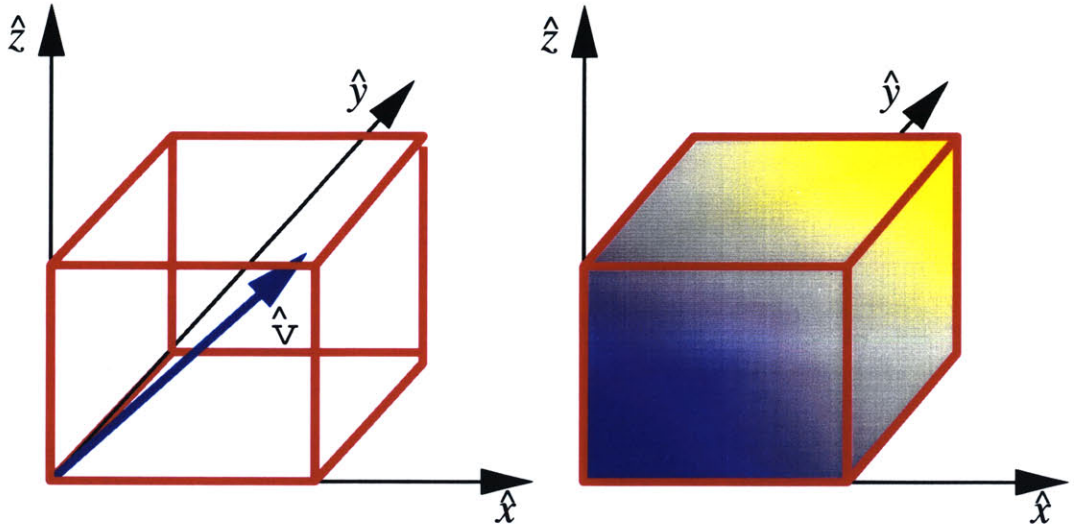


Figure 5-8: Intended designs for material distribution ($\mathbf{m}^*(\mathbf{x})$) and modeled compositions ($\mathbf{m}(\mathbf{x})$). $\delta_x, \delta_y, \delta_z = a$ (a.) Boundary of material feature lies along voxel boundaries. (b.) Boundary of material feature lies off voxel boundaries but is still captured in representation. (c.) Voxel mesh is too coarse to capture material feature.

5.5.2 Constraint based on gradient

Although a constraint on voxel size due to material accuracy when discontinuities exist is not possible, the design of regions of smoothly graded compositions ($\vec{\nabla}\mathbf{m}^*(\mathbf{x})$ is defined for all $\mathbf{x}$) does permit the definition of such a constraint. In this case, since voxels can represent only regions of uniform material, the error in composition is a function of the gradient in composition of the design function (or how quickly the desired volume fraction of each material is varying) within a region.



(a) (b)

Figure 5-9: (a) Region of linearly graded material and direction ($\hat{\mathbf{v}}$) of grading. (b) The desired graded to be assigned to the region.

For analysis purposes, we will begin by assuming the designer wishes to create a region of linearly graded composition at a rate of $\mathbf{M}^*$ along the direction $\hat{\mathbf{v}} = \frac{\sqrt{3}}{3}(1,1,1)$ (see Figure 5-9). Over a single voxel, the intended variation for each material (m_l^* for $0 \leq l < d_m$) can be expressed as:

$$m_l^*(\mathbf{x}) = m_l^*\Big|_A + \frac{\sqrt{3}M_l^*}{3}(x - x_i + y - y_j + z - z_k) \tag{5.13}$$

where $m_l^*\Big|_A$ is the intended volume fraction for material l at the lower corner (x_i, y_j, z_k) of the voxel v_{ijk}. The above equation represents the desired, ideal grading within the single voxel (see Figure 5-10a). The voxel, however, can only represent a single intensity level for each material resulting in an approximation error between the designed and modeled composition (see Figure 5-10b). If Algorithm 2 is used to assign the composition within the voxel, the intensity level for each material

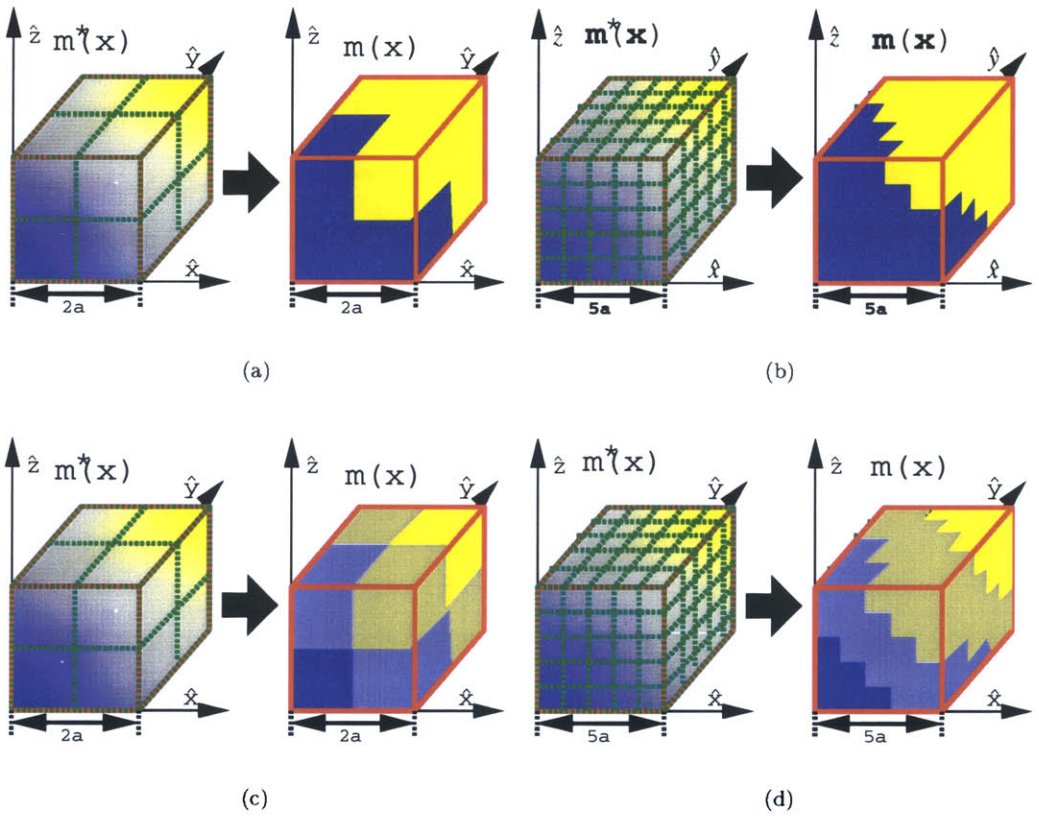


Figure 5-10: Examples of linearly graded designs thresholded to uniform composition assignments to voxels.

is set to its average value over the voxel:

$$\begin{aligned} \nu_l &= \frac{\int_{x_i}^{x_i+a}\int_{y_j}^{y_j+a}\int_{z_k}^{z_k+a}\mathbf{m}_l^*(x,y,z)dxdydz}{a^3} \\ &= m_l^*\Big|_A + \frac{\sqrt{3}}{2}M_l^* a \end{aligned}$$

As before, since a voxel can only represent discrete values, this average composition is thresholded as it is assigned to the voxel:

$$v_{ijk}.m_l = \frac{\left\lfloor \left(m_l^*\Big|_A + \frac{\sqrt{3}}{2}M_l^* a\right)(n_\lambda - 1) + \frac{1}{2} \right\rfloor - 1}{n_\lambda} \tag{5.14}$$

To determine a constraint on the voxel size, let us express the error for each material as the maximum difference between the intended volume fraction and the value actually assigned to each of the materials:

$$\epsilon_m = \max\left\{ m_l^*(\mathbf{x}) - v_{ijk}.m_l \middle| l \in [0, d_m), \mathbf{x} \in \{[x_i, x_i+a), [y_j, y_j+a), [z_k, z_k+a)\} \right\} \tag{5.15}$$

Referring to Equation 5.14, the maximum error will occur at the points (x_i, y_j, z_k) and $(x_i + a, y_j + a, z_k + A)$. For this grading, the composition error can then be expressed as:

$$\epsilon_m = \frac{\sqrt{3}a}{2} \max\left\{M_0^*, M_1^*, \ldots M_{d_m-1}^*\right\}. \tag{5.16}$$

Rearranging the above equation, we can express a constraint on the maximum voxel size as a function of composition gradient and composition accuracy:

$$a = \frac{2\sqrt{3}\epsilon_m}{3\max\left\{M_0^*, M_1^*, \ldots M_{d_m-1}^*\right\}} \tag{5.17}$$

The above constraint for voxel size was derived assuming a constant grading along the diagonal of the lattice of voxels. Figure 5-10 illustrates the relationship between voxel size, rate of composition variation, and voxel assignment for a couple of cases of linear grading. In general cases, the grading may vary and occur in any direction. As long as the intended grading, however, is subject to the constraint that the variation is continuous throughout the object, a prescribed material accuracy

(ϵ_m) can be imposed. This desired accuracy dictates the maximum dimension of the voxels:

$$\begin{aligned}
\delta_x &\leq \frac{2\sqrt{3}\epsilon_m}{3\max\left\{\vec{\nabla}m_0^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \vec{\nabla}m_1^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \ldots \vec{\nabla}m_{d_m-1}^*(\mathbf{x})\cdot\hat{\mathbf{v}}\right\}} \\
\delta_y &\leq \frac{2\sqrt{3}\epsilon_m}{3\max\left\{\vec{\nabla}m_0^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \vec{\nabla}m_1^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \ldots \vec{\nabla}m_{d_m-1}^*(\mathbf{x})\cdot\hat{\mathbf{v}}\right\}} \\
\delta_z &\leq \frac{2\sqrt{3}\epsilon_m}{3\max\left\{\vec{\nabla}m_0^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \vec{\nabla}m_1^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \ldots \vec{\nabla}m_{d_m-1}^*(\mathbf{x})\cdot\hat{\mathbf{v}}\right\}}
\end{aligned} \tag{5.18}$$

where $\mathbf{x}$ is any point in the object, $\mathbf{m}^*(\mathbf{x})$ is the intended, continuous grading, and $\mathbf{v}$ is any unit vector in $\mathcal{R}^3$. The denominator in Equation 5.18 represents the greatest rate of change in volume fraction of any material along any direction $\hat{\mathbf{v}}$.

5.6 Discussion

This chapter identified the factors affecting the storage costs for the exhaustive enumeration approach to modeling FGM objects. These factors include the geometric and material accuracies as well as the physical size of the modeled object and nature of the composition variation. The storage cost for the approach to implementing the exhaustive enumeration modeling scheme presented in Section 4.3, however, was given in terms of the number of voxels and the number of levels of material intensities represented by each voxel.

$$\begin{aligned}
S_{vox} &= S_{vox}[n_{vox}, n_\lambda] \\
&= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{1}{8}n_{vox}d_m\lceil \lg n_\lambda \rceil
\end{aligned}$$

Using the results from this chapter, the voxel size is a function of the geometric and material accuracies, as well as the maximum rate of material variation within the intended design. The number of voxels, in turn, can be determined from the dimensions of the model.

$$\begin{aligned}
\delta_x = \delta_y = \delta_z &= \delta\left(\epsilon_g, \epsilon_m, M^*\right) \\
&= \min\{\epsilon_g, \frac{2\sqrt{3}\epsilon_m}{3M^*}\} \qquad (5.19) \\
n_{vox} &= n_{vox}[L_x, L_y, L_z] \\
&= \left\lceil\frac{L_x}{\delta_x}\right\rceil\left\lceil\frac{L_y}{\delta_y}\right\rceil\left\lceil\frac{L_z}{\delta_z}\right\rceil \qquad (5.20) \\
&\quad \text{where } M^* = \max\left\{\vec{\nabla}m_0^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \vec{\nabla}m_1^*(\mathbf{x})\cdot\hat{\mathbf{v}}, \ldots \vec{\nabla}m_{d_m-1}^*(\mathbf{x})\cdot\hat{\mathbf{v}}\right\}
\end{aligned}$$

Therefore, the parameters affecting the storage cost of the voxelized method described in the previous chapter are the size of the object, the geometric and material accuracies, and maximum rate of

composition variation:

$$\begin{aligned} S_{vox} &= S_{vox}[L_x, L_y, L_z, a, \epsilon_m, d_m] \\ &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{1}{8} d_m \left\lceil \frac{L_x}{a} \right\rceil \left\lceil \frac{L_y}{a} \right\rceil \left\lceil \frac{L_z}{a} \right\rceil \left\lceil \lg \left(\frac{1}{2\epsilon_m} + 1 \right) \right\rceil \end{aligned} \tag{5.21}$$

$$\text{where } a = \min\{\epsilon_g, \mu_g, \frac{2\sqrt{3}\epsilon_m}{3M^*}, \mu_m\} \text{ and}$$

$$M^* = \max\left\{ \left| \vec{\nabla} m_0^*(\mathbf{x}) \cdot \hat{\mathbf{v}} \right|, \left| \vec{\nabla} m_1^*(\mathbf{x}) \cdot \hat{\mathbf{v}} \right|, \ldots \left| \vec{\nabla} m_{d_m-1}^*(\mathbf{x}) \cdot \hat{\mathbf{v}} \right| \right\}$$

The above expression only holds when the desired composition ($\mathbf{m}^*(\mathbf{x})$) is continuous. When discontinuities exist, M^* is undefined. Therefore, the concept of a material accuracy is not an appropriate parameter in such cases, only the geometric tolerance which can be applied to boundaries at the surface of the object and between the discontinuous region of different composition. The resolution (n_λ) in such cases becomes an independent variable:

$$\begin{aligned} S_{vox} &= S_{vox}[L_x, L_y, L_z, \epsilon_g, n_\lambda, d_m] \\ &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{1}{8} d_m \left\lceil \frac{L_x}{\epsilon_g} \right\rceil \left\lceil \frac{L_y}{\epsilon_g} \right\rceil \left\lceil \frac{L_z}{\epsilon_g} \right\rceil \left\lceil \lg n_\lambda \right\rceil \end{aligned} \tag{5.22}$$

The preceding equations relate the storage cost to the intentions of the designer. If the SFF manufacturing process is a constraint, the resolution at which the final product will be rendered is fixed. Therefore, another interesting question to ask is how much memory is required to store an object of a given size to be fabricated through a process with a given resolution. Instead of trying to reproduce the designer's intention to a prescribed resolution, restrict the resolution to that of the machine since any more information would just be lost during the fabrication process. In this case, the number of threshold levels each voxel would have to maintain is two ($n_\lambda = 2$) for each material, indicating the presence or not of a primitive of each material at the corresponding grid point in the lattice. Figure 5-11 is a plot of the memory requirements as a function of non-dimensional Build Space volume (V^*). The non-dimensional Build Space volume is the volume of region in which an SFF machine fabricates an object ($L_x \times L_y \times L_z$) normalized by the volume of a material primitive ($\delta_x \times \delta_y \times \delta_z$). Curves are plotted for material systems of varying dimensions. Figure 5-12 relates the storage growth to the Build Space volume for a process with a material primitive resolution of $\delta_x = \delta_y = \delta_z = 10^{-4}m$. The final figure, Figure 5-13, provides the storage cost for representing a Build Space with a cross-sectional area of $L_x \times L_z = 10^{-2}$ m^2 and the same primitive dimensions as a function of Build Space height (L_z).

In closing, it is important to note that the analysis here assumes that memory is allocated to explicitly represent each voxel in the lattice, as described in the previous chapter. Compression methods, such as run-length-encoding and octree data structures, could potentially be used to reduce the storage cost, at the risk of increasing the complexity of implementation and algorithms for

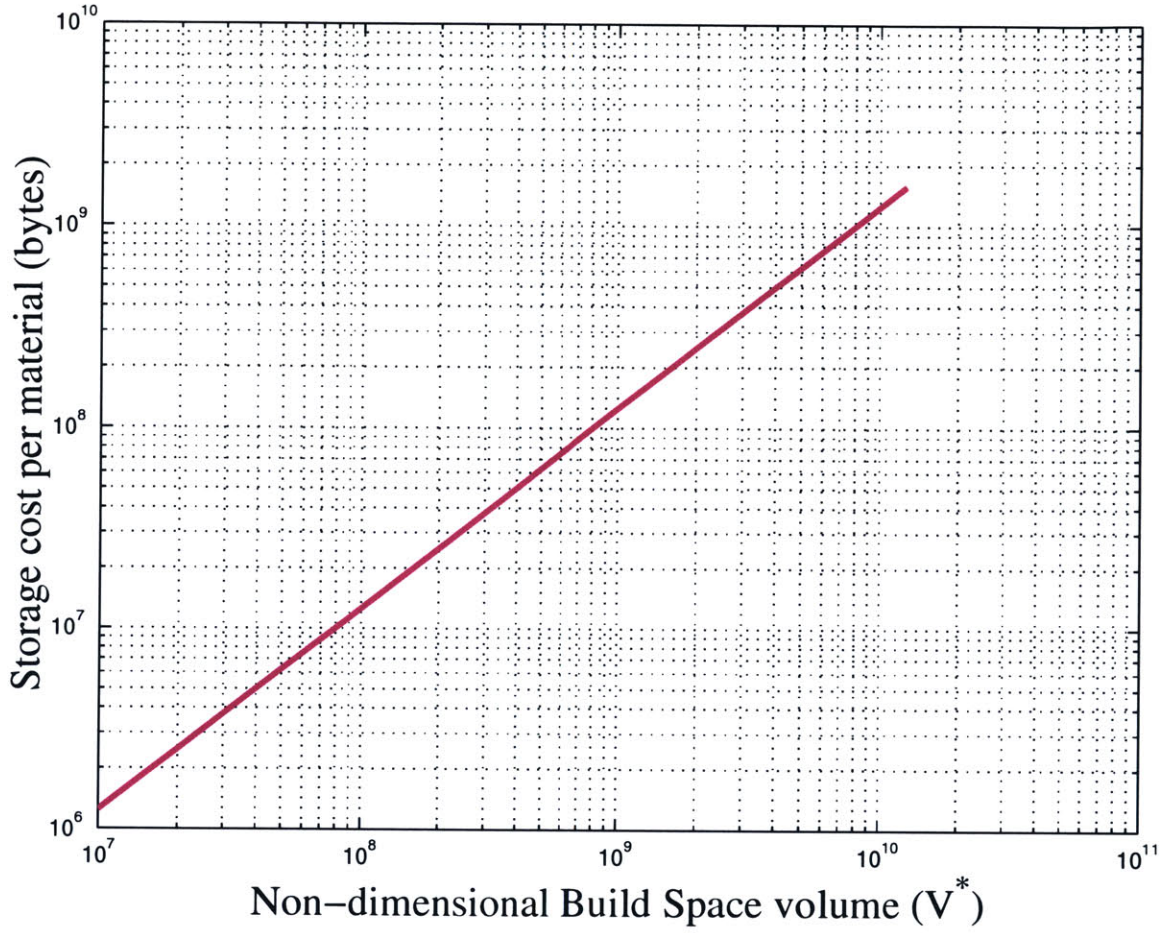


Figure 5-11: Storage cost (in bytes per material) required to represent a model as a function of normalized Build Space volume ($V^* = \frac{L_x \times L_y \times L_z}{\delta_x \times \delta_y \times \delta_z}$), which is equivalent to the number of voxels (n_{vox}). (slope = $\frac{1}{8}$ bytes)

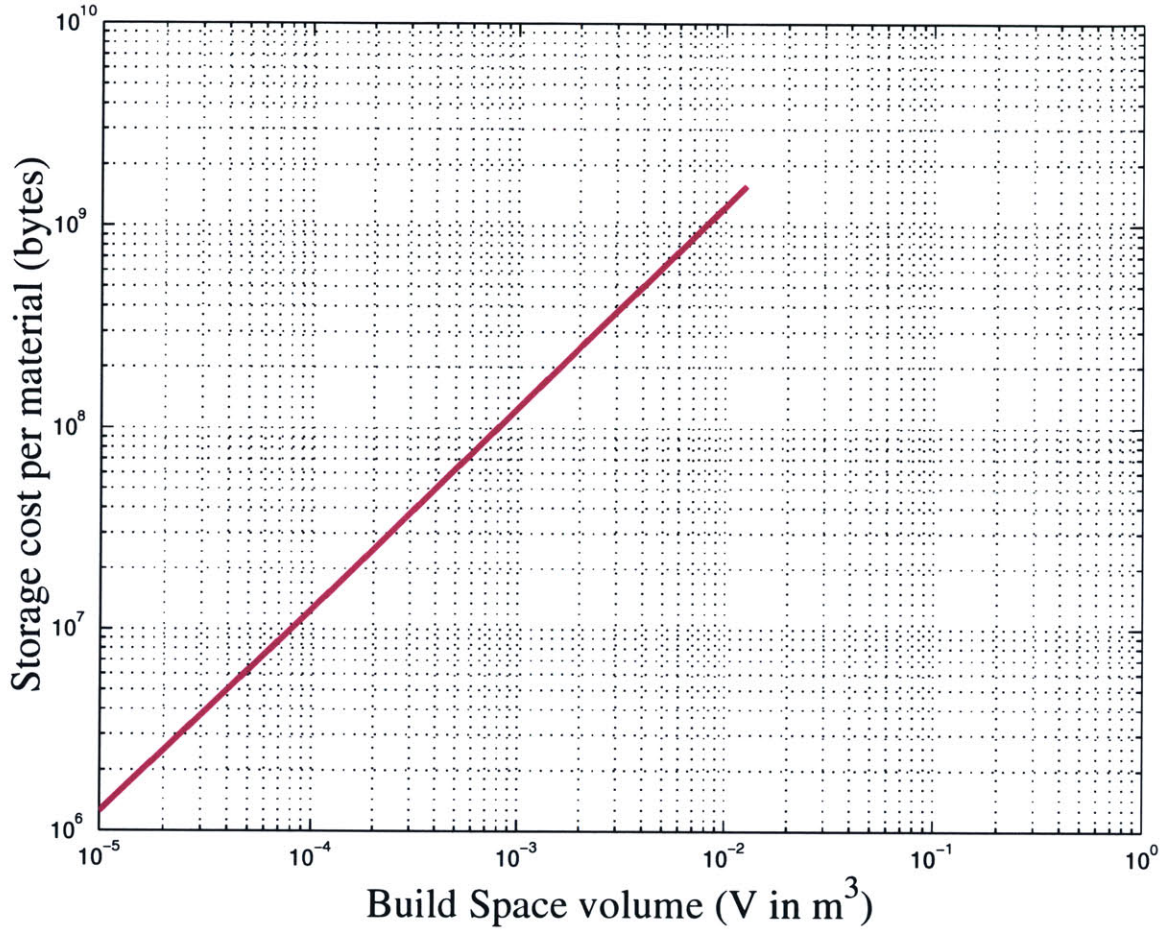


Figure 5-12: Storage cost (in bytes per material) required to represent a model as a function of Build Space volume (V in m^3) for an SFF process with a resolution of $\delta_x = \delta_y = \delta_z = 10^{-4}m$. (slope = $1.25 \times 10^{11} \frac{\text{bytes}}{m^3}$)

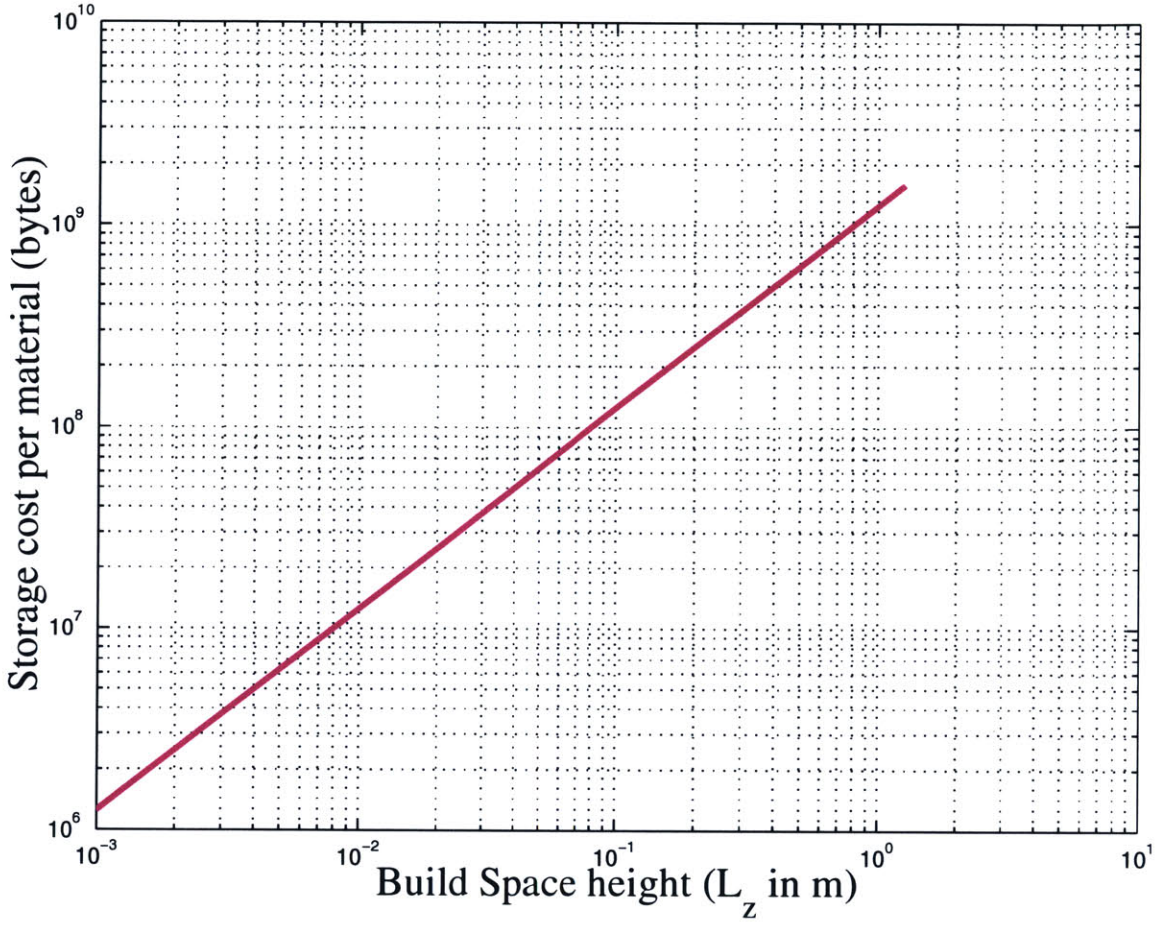


Figure 5-13: Storage cost (in bytes per material) required to represent a Build Space with a cross-sectional area of $L_x \times L_y = 10^{-2}m^2$ as a function of Build Space height (L_z), for an SFF process with a resolution of $\delta_x = \delta_y = \delta_z = 10^{-4}m$ (slope = $1.25 \times 10^9 \frac{\text{bytes}}{m}$).

designing composition. These approaches are not addressed in this thesis.

Chapter 6

Bounds for meshed model growth

6.1 Motivation

Meshed schemes for representing solid models require that models be decomposed into sets of nodes (points in space) and elements that interpolate these nodes. This requirement limits the scope of models that can be represented exactly to only those that can be decomposed precisely into the finite elements, thereby limiting the geometric and composition variations that can be represented. Linear finite elements, for instance, are capable of representing only planar faceted models with linear variation in composition with 100% accuracy. In the practice of Finite Element Analysis, however, the shapes of higher order (curved) models are regularly approximated by large numbers of smaller, lower order elements. The number of elements required for an analysis is determined by how many are needed to provide a converged solution for the numerical simulation. For representing FGM objects, the same approach can be taken, in which models are decomposed into meshes of finite elements. For this purpose, however, the goal is not to find a converged solution for some analysis, but to provide an accurate representation of some intended design, both in terms of geometry and composition. This chapter examines the criteria for subdividing models into finite elements and provides bounds for the rate of growth of the storage requirements for such models.

6.2 Curve meshing

Before exploring issues in surface and volume meshing, the number of straight line segments that are needed to approximate a curve in space is investigated. First, the number of line segments to approximate a circular arc within a given geometric accuracy (ϵ_g) is derived. This relationship is then used to determine an upper bound for the number a linear segments necessary to approximate any space curve as a function of its curvature (κ_g).

6.2.1 Approximating a circular arc

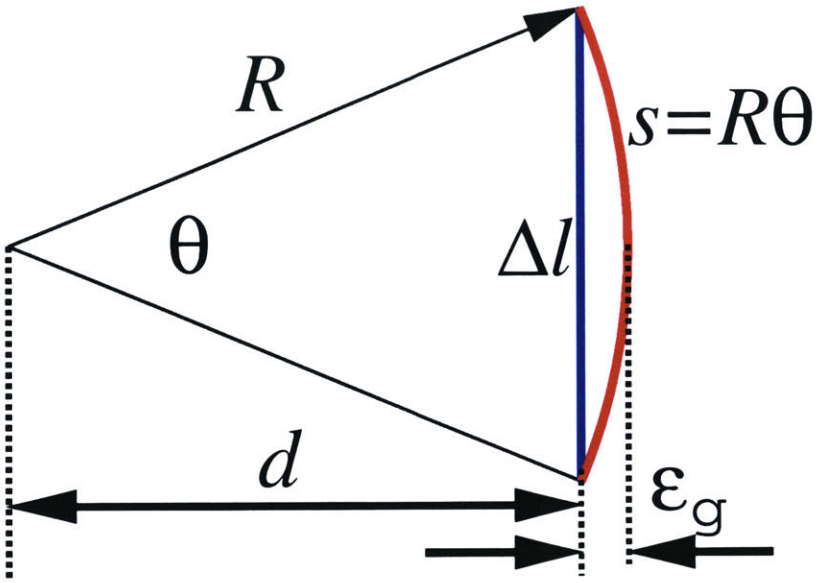


Figure 6-1: Error associated with approximating a circular arc with a straight line segment.

Consider the approximation of a circular arc of radius R with a chain of line segments. The approximation error is defined as the maximum distance between any of the line segments and the corresponding arc (see Figure 6-1). Let ϵ_g represent this error and d be the distance from the arc's center of curvature and the line segment. Then

$$\epsilon_g = R - d \tag{6.1}$$

The length of the line segment (Δl) can be related to the approximation error and the arc's radius using simple trigonometry:

$$\begin{aligned} \epsilon_g &= R - \frac{\sqrt{4R^2 - (\Delta l)^2}}{2} \\ \Delta l &= 2\sqrt{\epsilon_g(2R - \epsilon_g)}, \end{aligned} \tag{6.2}$$

If Δs is the arclength of a section of the arc that can be accurately approximated by a single line segment, the length of corresponding line segment is:

$$\begin{aligned} \Delta l &= 2R \sin \frac{\theta}{2} \\ &= 2R \sin \frac{\Delta s}{2R} \end{aligned} \tag{6.3}$$

Solving for the arclength from the above equations, an expression for the length of the largest arc that can be approximated by a straight line segment while remaining within a prescribed accuracy

is formed:

$$\Delta s = 2R \arcsin\left(\frac{\sqrt{\epsilon_g(2R-\epsilon_g)}}{R}\right). \tag{6.4}$$

Using the bounds derived by Filip *al* [18], the maximum arclength of a circular arc that can be approximated by a single line segment within a prescribed accuracy ϵ_g is

$$\Delta s_{filip} = 2\sqrt{2\epsilon_g R}. \tag{6.5}$$

For small ϵ_g (more precisely $\epsilon_g \ll 2R$), the higher order terms in Equation 6.4 can be ignored and the two expressions have the same leading order behavior:

$$\begin{aligned} O(\Delta s) &= 2\sqrt{2\epsilon_g R} \\ &= \Delta s_{filip} \end{aligned}$$

The variation of arclength with the prescribed accuracy is plotted in Figure 6-2.

The expresion derived here (Equation 6.4) includes higher order terms and is exact for a circular arc. By choosing to use Equation 6.4 instead of Equation 6.5, a tighter bound on the number of line segments required to approximate a circular arc (as a function of arclength s) is formed:

$$n_{segments} = \left\lceil \frac{s}{\Delta s} \right\rceil \tag{6.6}$$

$$= \left\lceil \frac{s}{2R \arcsin\left(\frac{\sqrt{\epsilon_g(2R-\epsilon_g)}}{R}\right)} \right\rceil \tag{6.7}$$

The convergence of the chain of straight line segments to the desired geometry of a circular arc is illustrated in Figure 6-3. The number of line segments required to approximate a given circular arc of unit arclength is plotted as a function of $\frac{R}{\epsilon_g}$ in Figure 6-4.

6.2.2 Approximating a G^1 curve

For a curve that is tangent (G^1) continuous (see Figure 6-5(a)), the previously derived relationship can be used to determine an upper bound for the number of line segments needed to accurately approximate the curve. Replacing the radius of the arc with the minimum radius of curvature (which corresponds to $\kappa_{g,max}$) of the more general curve ($R \leftarrow \frac{1}{\kappa_{g,max}}$), the maximum arclength of

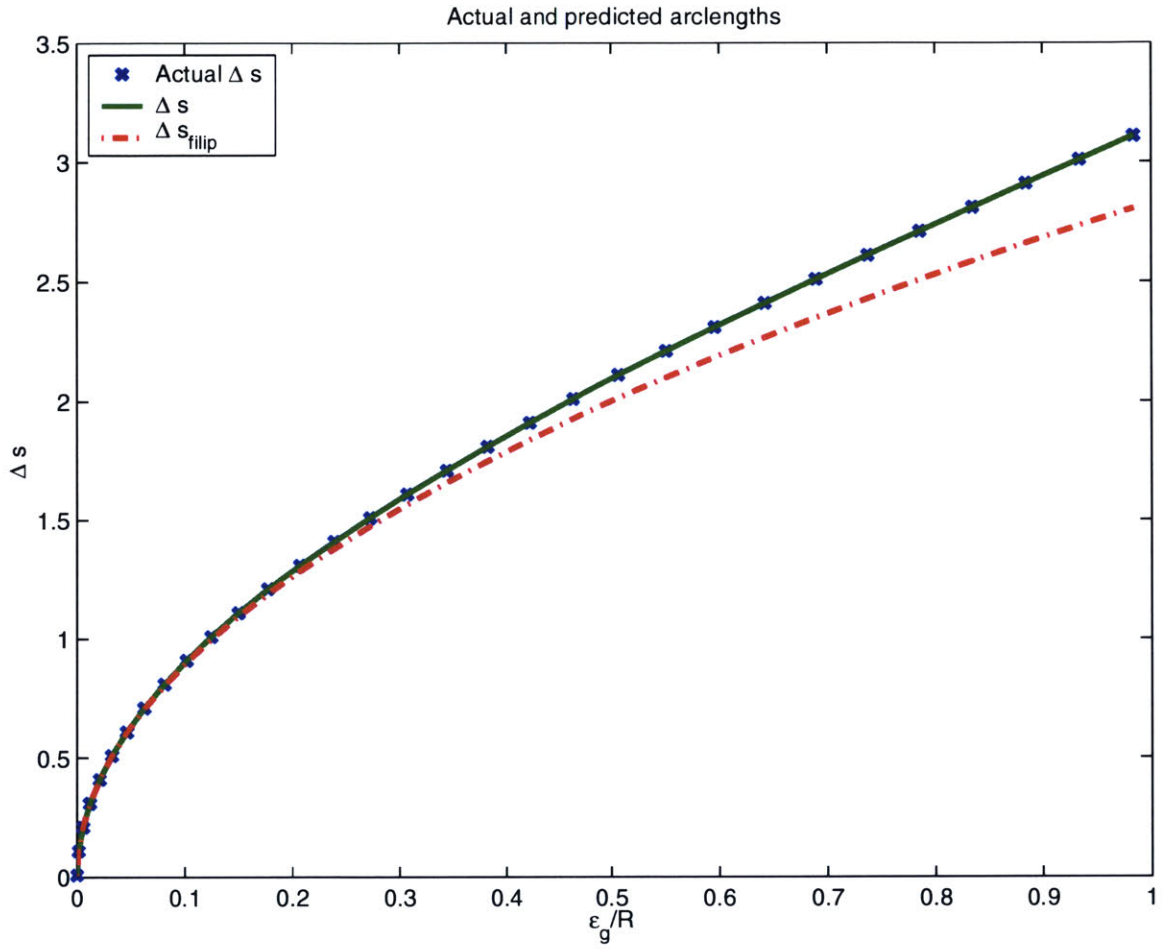
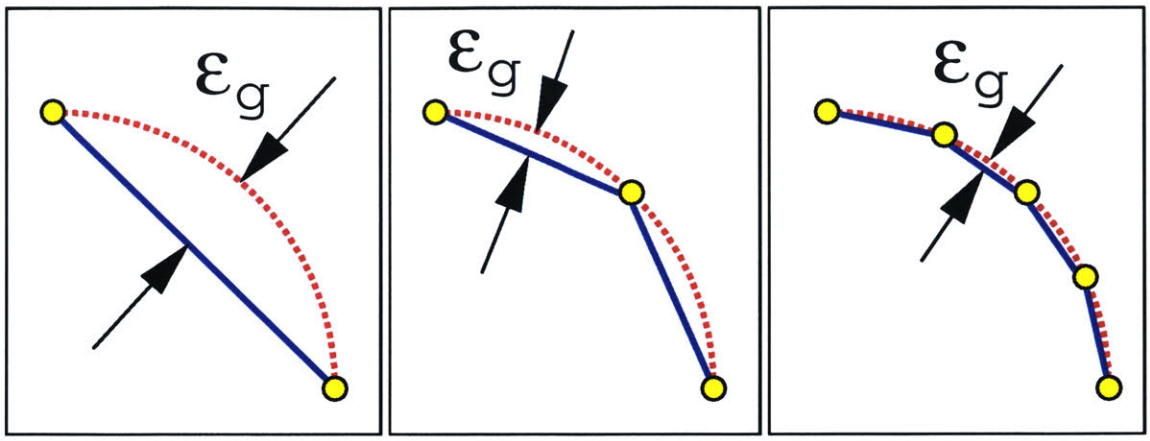


Figure 6-2: The maximum arclength of a circular arc (Δs) that can be approximated by a single line segment within a prescribed accuracy (ϵ_g), plotted as a function of $\frac{\epsilon_g}{R}$. The actual, maximum arclength that can be approximated by a single line is also plotted.



(a) (b) (c)

Figure 6-3: Convergence to the shape of the arc with an increasing number of straight line segments: (a) $n_{segments} = 1$, (b) $n_{segments} = 2$, and (c) $n_{segments} = 4$.

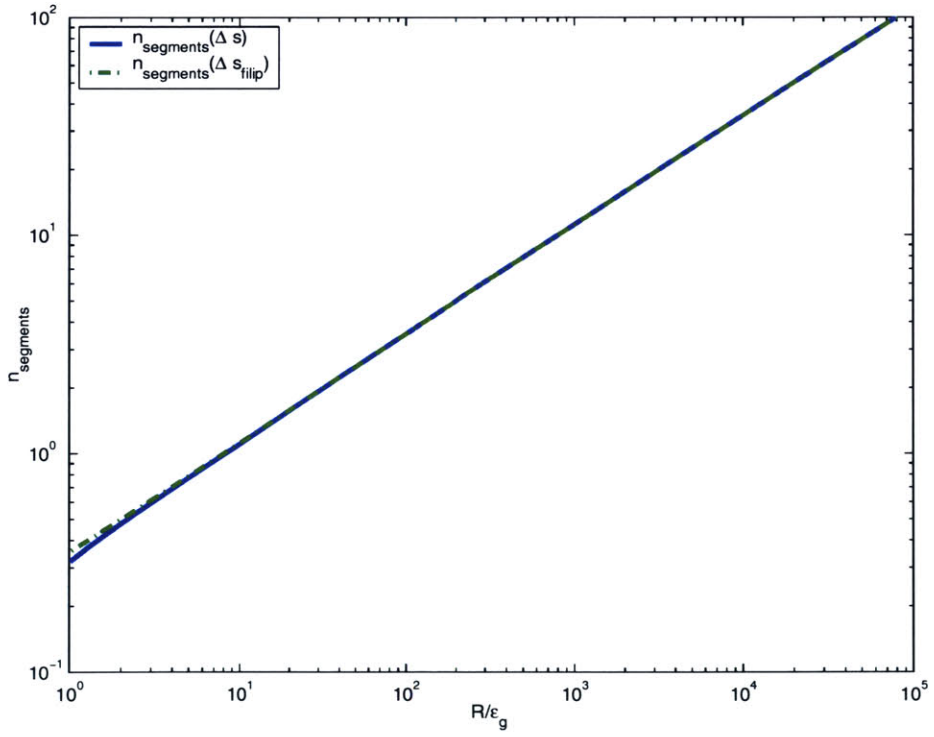
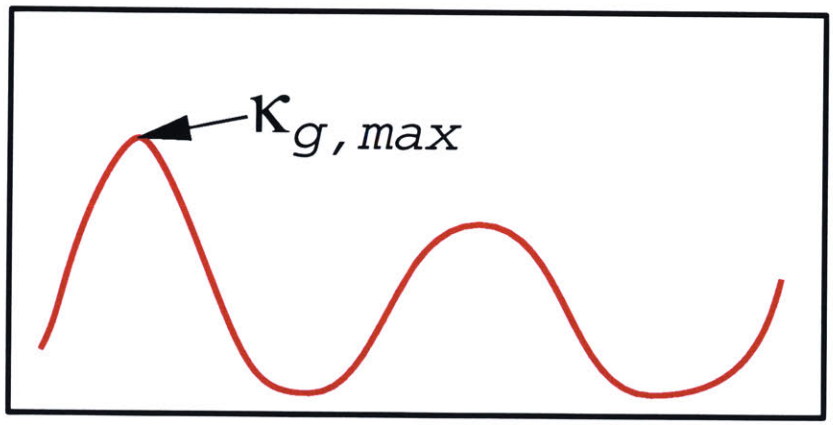
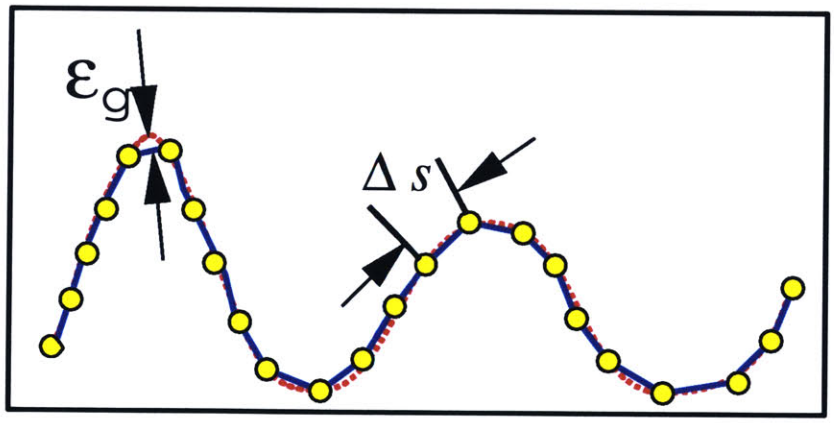


Figure 6-4: The number of line segments required to approximate a circular arc of unit arclength, plotted as a function $\frac{R}{\epsilon_g}$.



(a)



(b)

Figure 6-5: (a) An arbitrary tangent continuous curve. (b) Approximation of a tangent continuous curve with a chain of line segments.

any G^1 curve that can be approximated with a straight line segment for a given accuracy is found:

$$\Delta s = \begin{cases} \frac{2\arcsin\left(\sqrt{\epsilon_g \kappa_{g,max}(2-\epsilon_g \kappa_{g,max})}\right)}{\kappa_{g,max}} & \text{for } \kappa_{g,max} \neq 0, \\ s \text{ otherwise.} \end{cases} \tag{6.8}$$

Therefore, the total number of line segments required to approximate an arbitrary tangent continuous curve as a function of its total arclength and maximum curvature, is:

$$n_{segments} = \left\lceil \frac{s}{\Delta s} \right\rceil \text{ (where s is the arclength of the curve)} \tag{6.9}$$

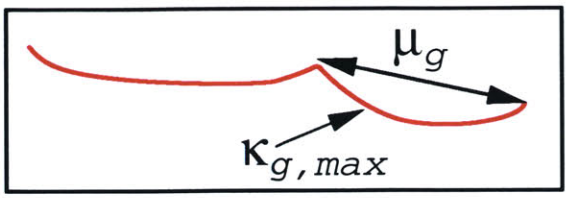
$$= \begin{cases} \left\lceil \frac{s\kappa_{g,max}}{2\arcsin\left(\sqrt{\epsilon_g \kappa_{g,max}(2-\epsilon_g \kappa_{g,max})}\right)} \right\rceil & \text{for } \kappa_{g,max} \neq 0, \\ 1 \text{ otherwise.} \end{cases} \tag{6.10}$$

This expression gives an upper bound for the number of approximating straight line segments as a function of the maximum curvature of the curve (or the length of the curve, in case the curve is linear), as shown in Figure 6-5(b). Since the curvature is not required to be constant, an adaptive subdivision scheme could be developed in which areas of less curvature are approximated by longer segments, thereby reducing the number segments needed to remain within tolerance. The focus here, however, is to provide an upper bound on the number of subdivisions, not to propose algorithms for adaptive curve approximation. Such algorithms can be found in Wolter and Tuohy [76] and Cho *et al* [12].

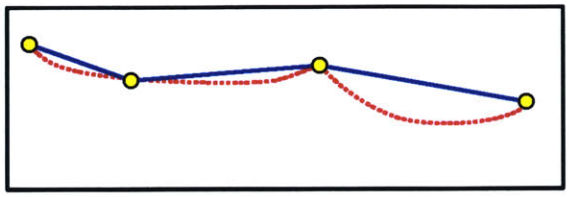
6.2.3 Approximating an arbitrary curve or curves

The above expression provides the number of equal length line segments required to guarantee that the desired G^1 curve is approximated within a specified tolerance. For cases in which the curve contains discontinuities of the tangent vector, as shown in Figure 6-6(a), the concept of a minimum geometric feature size (the distance between between tangent discontinuities) may impose a limit on the segment size. If the distance between two points of tangent discontinuity (μ_g) is less than the segment length found based on the curve's curvature, this feature size becomes the limiting constraint in the size of the line segment, as shown in Figure 6-6(b).

$$\Delta s = \min\left\{ \frac{2\arcsin\left(\sqrt{\epsilon_g \kappa_{g,max}(2-\epsilon_g \kappa_{g,max})}\right)}{\kappa_{g,max}}, \mu_g \right\} \tag{6.11}$$



(a)



(b)

Figure 6-6: (a) An arbitrary curve. (b) Approximation of an arbitrary curve with a chain of line segments.

And the total number of line segments is

$$n_{segments} = \max\left\{\left\lceil \frac{s\kappa_{g,max}}{2\arcsin\left(\sqrt{\epsilon_g \kappa_{g,max}(2-\epsilon_g\kappa_{g,max})}\right)}\right\rceil, \left\lceil \frac{s}{\mu_g}\right\rceil\right\}. \tag{6.12}$$

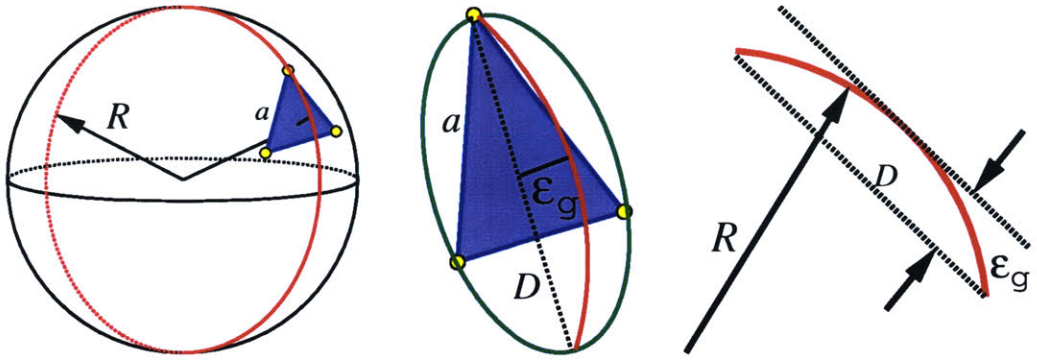
6.3 Surface meshing

In finite element meshes, surfaces are decomposed into simpler elements, such as triangles and quadrilaterals. In this section, only triangular meshes are considered for simplicity in analysis but the trends and concepts are equally applicable to other types of elements. The upper bound on the rate of growth of the triangular mesh that will be developed here will be applicable to any tangent plane continuous surface.

6.3.1 Approximating the surface of a sphere

To begin the analysis of the growth of surface meshes, the approximation of a spherical patch with a mesh of equilateral triangles is first considered. If the desired patch has a surface area of $A_{surface}$, the number of triangles of equal area required to tessellate the surface is simply:

$$n_{triangles} = O\left\lceil \frac{A_{surface}}{A_{triangle}}\right\rceil \tag{6.13}$$



(a) (b) (c)

Figure 6-7: (a) A triangle approximating a patch of a sphere's boundary. (b) Enlarged view of triangle showing the circumscribed circle and the approximation error. (c) Approximation of circular arc of radius R with a line segment of length D.

For an equilateral triangle, its area is a function of the length of its sides, a:

$$A_{triangle} = \frac{\sqrt{3}a^2}{4} \tag{6.14}$$

From the preceding analysis for the curve, an expression was formed for the maximum length of a line segment to approximate a circular arc within a prescribed accuracy (see Equation 6.4). A similar analysis can be performed here. Consider an equilateral triangle whose three vertices lie on the surface of a sphere, as shown in Figure 6-7(a). The maximum distance between this triangle and the sphere occurs at the centroid of the triangle, the point where the radius of the sphere is normal to the triangle. This is the same error as would result from the approximation of a portion of a great arc about the sphere with the diameter of the triangle's circumscribed circle (see Figures 6-7(b) and (c)). By relating the diameter of the triangle's circumscribed circle (D) to the dimension of the triangle (a), and substituting Δs from Equation 6.4 for D, an expression for the size of the equilateral triangle as a function of geometric accuracy (ϵ_g) and radius (R) of the spherical patch is found.

$$\begin{aligned} D[R] &= D[a] \\ 2R \arcsin\left(\frac{\sqrt{\epsilon_g(2R-\epsilon_g)}}{R}\right) &= \frac{2\sqrt{3}a}{3} \\ \Rightarrow a &= \sqrt{3}R \arcsin\left(\frac{\sqrt{\epsilon_g(2R-\epsilon_g)}}{R}\right) \end{aligned} \tag{6.15}$$

Substituting the above expression for a in into Equations 6.14- 6.13, the number of triangles required

to approximate a spherical patch is found.

$$\begin{aligned} n_{triangles} &= O\left\lceil \frac{A_{surface}}{A_{triangle}} \right\rceil \\ &= O\left\lceil \frac{4\sqrt{3}A_{surface}}{3a^2} \right\rceil \\ &= O\left\lceil \frac{4\sqrt{3}A_{surface}}{9R^2\left[\arcsin\left(\frac{\sqrt{\epsilon_g(R-\epsilon_g)}}{R}\right)\right]^2} \right\rceil \end{aligned} \tag{6.16}$$

Note that the above derivation (Equation 6.16) does not take into account the nature of the boundary curve of the patch; the assumption made here is that the limiting factor for the dimension of an edge of the triangle a is the radius of the sphere R, not the curvature or miminum feature size of the patch's boundary curve(s).

To mesh an entire sphere of radius R ($A_{surface} = 4\pi R^2$), the number of required equilateral triangles is:

$$n_{triangles} = O\left\lceil \frac{16\sqrt{3}\pi}{9\left[\arcsin\left(\frac{\sqrt{\epsilon_g(2R-\epsilon_g)}}{R}\right)\right]^2} \right\rceil \tag{6.17}$$

Numerical confirmation of this expression is given in Figure 6-8, in which the predicted number of triangles required to mesh a unit sphere is plotted with the number of triangles generated by triangulation of a sphere on a commercial CAD system (SolidWorks^{TM}).

6.3.2 Approximating an arbitrary surface patch

As in the case of the free-form curve, an upper bound for the number of equilateral triangles required to accurately approximate a free-form surface patch can be formed and may be a function of two things [18]: minimum feature size and curvaure. The first restriction on triangle size is due to the minimum feature size (μ_g) of the surface patch or its boundary curve(s). The minimum feature size of the surface patch is evaluated as the minimum width of the patch or the distance between tangent plane discontinuities. The smallest feature of the patch's bounary curve(s) follows the analysis in the preceding section, and provides is also considered when determininig the minimum feature size for the surface (μ_g). The second factor affecting triangle size is maximum curvature ($\kappa_{g,max}$). The maximum curvature is either the maximum curvature of the absolute values of principle curvatures of the surface or the maximum curvature of its boundary curve(s). To form an expression for a bound on the number of triagnles required to approximate a free-form surface, Equation 6.16 is modified to take into account the maximum curvature of the free-form patch ($R \rightarrow \frac{1}{\kappa_{g,max}}$) and its

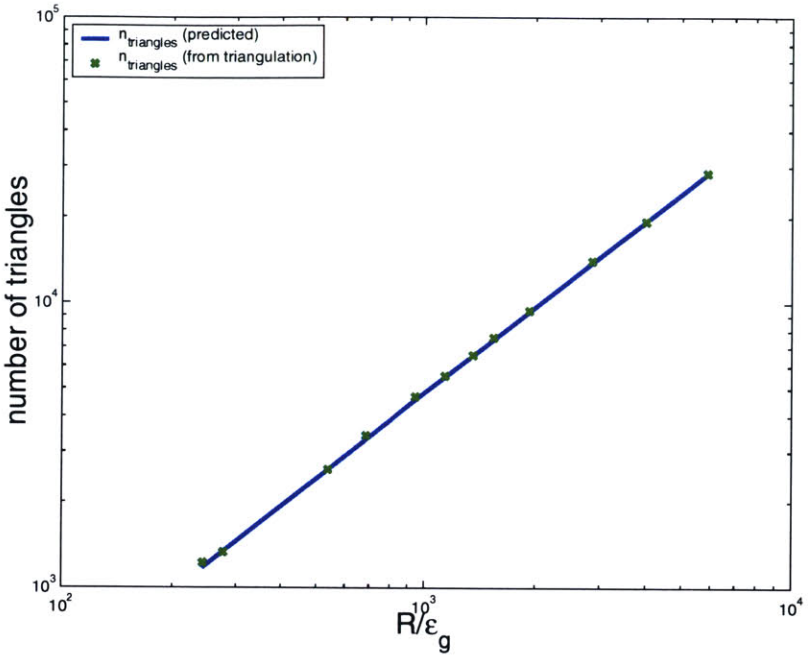


Figure 6-8: Number of triangles required to mesh a sphere to achieve a prescribed approximation accuracy. The data was generated from the STL export module in SolidWorks^{TM}.

mininum feature size,

$$n_{triangles} = O\left\lceil \frac{4\sqrt{3}A_{surface}}{3a^2} \right\rceil \tag{6.18}$$

$$\text{where } a = \min\left\{ \frac{\sqrt{3}}{\kappa_{g,max}} \left(\Delta s, \mu_g, \arcsin\sqrt{\epsilon_g \kappa_{g,max}(1 - \epsilon_g \kappa_{g,max})} \right) \right\}$$

As for the curve case, the above represents an upper bound to the case when then surface is meshed uniformly with equilateral triangles. For free-form surfaces with varying curvature, an adaptive meshing approach would yield fewer triangles, see Cho *et al* [11].

6.4 Volume meshing

The formulation for the upper bound on the number of tetrahedra required to accurately represent a model follows the same approach as the bounds for the curve and surface meshes. For analysis, let us assume that each tetrahedron is regular, composed of four equilateral triangles. The number of tetrahedra of dimension a in the mesh is simply:

$$n_{tetrahedra} = O\left\lceil \frac{V_{region}}{V_{tetrahedron}} \right\rceil \tag{6.19}$$

where V_{region} is the volume of the region to be meshed. The volume of the tetrahedron, as a function of the length of a side a, is:

$$V_{tetrahedron} = \frac{\sqrt{2}a^3}{12} \tag{6.20}$$

The length of a side of the tetrahedron is determined by two factors: the geometry of the region and the composition variation of the region. The first factor concerns the shape of the region's boundary and dictates the size of the triangles required to mesh the surface. This is explained in the preceding section. The second factor depends on the composition. Just as the geometric curvature (κ_g) or minimum geometric feature size (μ_g) determined the number of linear segments required to approximate a curve, a material curvature(κ_m) or minimum material feature size (μ_{mt}) will place an upper limit on the size of the tetrahedra in the mesh, as will be explained below.

6.4.1 Bounds on the number of tetrahedra due to geometric accuracy

The first factor to restrict the size of the tetrahedra is the shape of the boundary. As explained in the previous section, the minimum geometric feature size or maximum geometric curvature of the region's bounding faces places a restriction on the size of the triangles in a mesh that still guarantees that the boundary is approximated within a given tolerance. Assuming that the mesh is made entirely of regular tetrahedra of identical size, Equations 6.13, 6.14, 6.19, and 6.20 can be related to express the number of tetrahedra throughout the region as a function of the number of surface triangles:

$$\begin{aligned} n_{tetrahedra} &= O\left\lceil \left(\frac{3^{\frac{1}{4}}}{2^{\frac{3}{2}}}\right)\left(\frac{n_{tri}}{A_{mod}}\right)^{\frac{3}{2}}(V_{mod}) \right\rceil \\ &\approx O\left\lceil 2.418\left(\frac{n_{tri}}{A_{mod}}\right)^{\frac{3}{2}}(V_{mod}) \right\rceil \end{aligned} \tag{6.21}$$

where V_{mod} and A_{mod} are the volume of the model and its surface area, respectively. The above inequality provides an expression for the maximum number of regular tetrahedra to fill a region interior to a surface meshed with n_{tri} equilateral triangles. The number of triangles required to accurately approximate the region's boundary is determined according to the previous section for surface meshing.

6.4.2 Material curvature

The concept of material curvature (κ_m) quantifies how quickly the rates of variation in the volume fractions of the different materials are changing along a given direction. Consider the FGM block illustrated in Figure 6-9(a). To evaluate κ_m at a point $\mathbf{x}$ in some direction $\hat{\mathbf{v}}$ (see Figure 6-9(b)), a

parametric line is first defined such that $\mathbf{p}(u)|_{u=0} = \mathbf{x}$ and $\dot{\mathbf{p}}(u)| = \hat{\mathbf{v}}$:

$$\mathbf{p}(u) = \mathbf{x} + \hat{\mathbf{v}}u. \tag{6.22}$$

The variation of the volume fraction of each material along this line can expressed as a parametric curve in Material Space (see Figure 6-9(c)), as a function of parameter u:

$$\mathbf{m}^*(\mathbf{x})|_{\text{along } \mathbf{p}(u)} = \mathbf{m}^*[\mathbf{p}(u)] = \mathbf{m}^*(u) \tag{6.23}$$

Let the curvature [66] for each material be defined as:

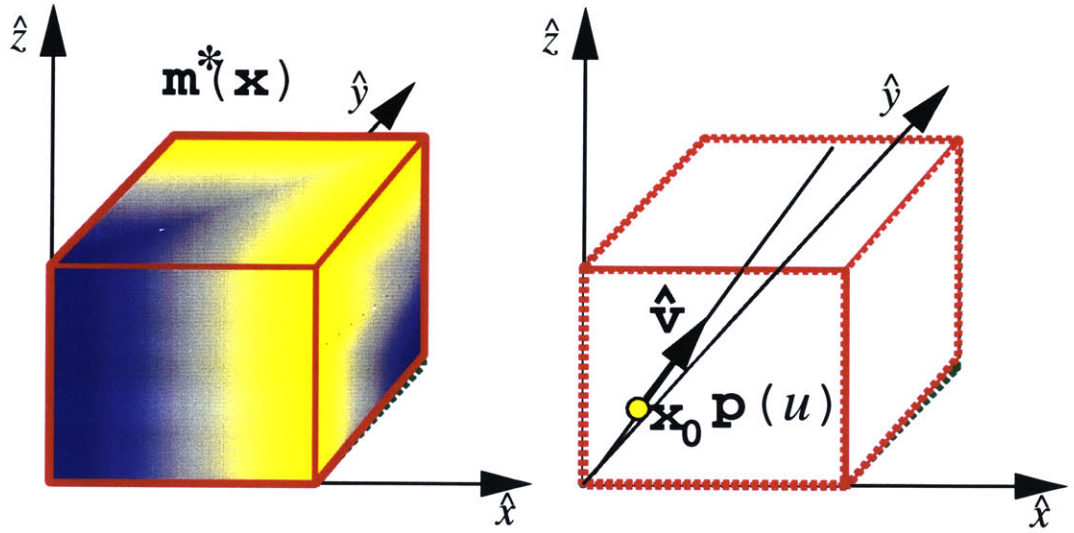
$$\begin{aligned} \kappa_{m_i} &= \frac{\ddot{m}_i^*(u)}{[1 + \dot{m}_i^*(u)^2]^{\frac{3}{2}}} \\ &= \frac{\vec{\nabla}[\vec{\nabla} m_i^*(\mathbf{x}) \cdot \hat{\mathbf{v}}] \cdot \hat{\mathbf{v}}}{\left[1 + \left(\vec{\nabla} m_i^*(\mathbf{x}) \cdot \hat{\mathbf{v}}\right)^2\right]^{\frac{3}{2}}}. \end{aligned} \tag{6.24}$$

The material curvature κ_m is a vector of values reflecting how quickly the rates of variation for each of the materials is changing along a given direction. As will be explained below, the parameter κ_m will become one of the factors limiting the maximum size of the tetrahedra in a mesh to guarantee a prescribed material accuracy.

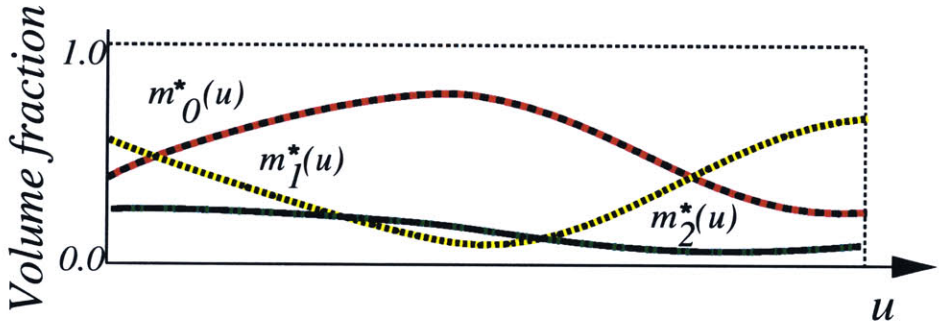
A more general approach for evaluating material curvature can be adopted from Nielson *et al* [48]. They extend the concept of Gaussian curvature from surfaces [66, 67] to trivariate functions. In their approach, the three principal curvatures (κ_1, κ_2, and κ_3) are evaluated as the eigenvalues of the following $3 \times 3 matrix$:

$$G = \frac{1}{1 + m_{i,x}^{*2} + m_{i,y}^{*2} + m_{i,z}^{*3}} \begin{pmatrix} m_{i,xx}^* & m_{i,xy}^* & m_{i,xz}^* \\ m_{i,yx}^* & m_{i,yy}^* & m_{i,yz}^* \\ m_{i,zx}^* & m_{i,zy}^* & m_{i,zz}^* \end{pmatrix} \begin{pmatrix} 1 + m_{i,xx}^{*2} & m_{i,x}^* m_{i,y}^* & m_{i,x}^* m_{i,z}^* \\ m_{i,x}^* m_{i,y}^* & 1 + m_{i,yy}^{*2} & m_{i,y}^* m_{i,z}^* \\ m_{i,x}^* m_{i,z}^* & m_{i,y}^* m_{i,z}^* & 1 + m_{i,zz}^{*2} \end{pmatrix}^{-1} \tag{6.25}$$

The maximum of the absolute values of the principle curvatures ($|\kappa_1|$, $|\kappa_2|$, and $|\kappa_3|$) would be the limiting curvature used in determining the tetrahedron size. The Nielson *al*'s work is introduced for completeness but is not explored since the goal here is to simply determining a limit on the tetrahedral size, not explore the differential properties of trivariate functions in depth. Extending methods for the interrogration of surfaces [68] to material compositions, however, is a promising direction for future research and should be explored in future work.



(a) (b)



(c)

Figure 6-9: (a) Hypothetical cube of graded material. (b) Parametric line $\mathbf{p}(u)$ through the graded material. (c) Hypothetical variations of the volume fractions of the different materials along the curve $\mathbf{p}(u)$.

6.4.3 Bounds on number of tetrahedra due to composition accuracy

Just as the intended shape for an object restricts the dimensions of the tetrahedra, the second factor affecting the size of the tetrahedra is the nature of the intended composition within the region. Similar to approximating the geometry of surfaces and curves, the tetrahedra must approximate the desired variation in composition, thus placing a restriction on the size of the tetrahedra in the mesh. The composition variation may limit the size of the tetrahedra in two ways: minimum material feature size (μ_{mt}) or the maximum material curvature (κ_m).

Restriction due to minimum material feature size

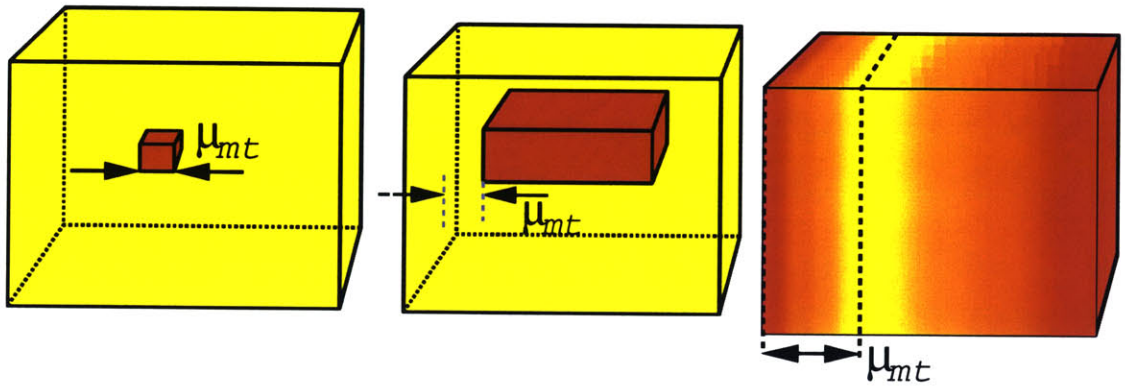


Figure 6-10: (a) Minimum material feature size determined by the minimum dimension of an internal feature of uniform composition. (b) Minimum material feature size determined by the minimum distance of the object's boundary and an internal feature of the uniform composition. (c) Minimum material feature determined by the distance between two discontinuities in material variation.

The first case arises from the desire to represent subregions of discontinuous material compositions or gradings within a model, as shown in Figure 6-10. Similar to the minium material feature size in the voxelized analysis, the minimum distance (μ_{mt}) between two material boundaries may be used as an upper limit for the size of the edges (a) in the mesh. In addition, the distance beween any point of discontinuity in the material derivative with another discontinuity (either a material boundary or derivitive) must also be considered (Figure 6-10(c)) in order for the mesh to capture the desired gradings. Substituting μ_{mt} into Equation an upper bound on the number of tetrahedra as a function of minimum material feature size is found from Equations 6.19,

$$\begin{aligned} n_{tetrahedra} &= O\left[\frac{V_{region}}{V_{tetrahedron}}\right] \\ &= O\left[\frac{12V_{region}}{\sqrt{2}\mu_{mt}^3}\right] = O\left[\frac{6\sqrt{2}V_{region}}{\mu_{mt}^3}\right] \end{aligned} \tag{6.26}$$

Restriction due to maximum material curvature

The second restriction applies to regions of smoothly varying composition, in which the composition may not only vary linearly but according to a higher order function. The nature of the intended variation places a restriction on the largest tetrahedron which can be used while still guaranteeing the modeled composition remains within a prescribed accuracy (ϵ_m) of the intended composition function ($\mathbf{m}^*(\mathbf{x})$). For the curve and surface cases, the analogous limitation was due to the curvature of the curve or surface, subject to the restriction that the geometry be tangent continuous. In this case, the intended material composition must vary smoothly, requiring that $\vec{\nabla}\mathbf{m}^*(\mathbf{x})$ be a continuous function of $\mathbf{x}$ for all materials, at all points within the region. Just as the curves and surfaces were required to be G^1 continuous in the preceding analyses, we can say that the FGM variation is required to be M^1 continuous.

For an M^1 continuous region of FGM material, the maximum material curvature of the desired material variation $\mathbf{m}^*(\mathbf{x})$ will dictate the maximum size of a tetrahedron that still guarantees the approximation of the desired composition within the prescribed accuracy ϵ_m. The derivation can be treated in the same way the maximum length of a line segment was determined in Equation 6.10 for approximating a G^1 curve. In Figure 6-9(a) and Equation 6.22, the material variation in a block was formulated along a line passing through point $\mathbf{x}$ as a function of parameter u, as the parametric curve $\mathbf{p}(u)$ in Material Space. Following Equation 6.8, the maximum size of uniform parametric subdivisions that still guarantee the approximation of the desired function within a specified accuracy can be found for each material, using the curvature for each material. If the maximum material curvature is known (for all $\mathbf{x}$, $|\hat{\mathbf{v}}| = 1$, and $m_i^*(x) \in \mathbf{m}^*(\mathbf{x})$), the maximum length for a tetrahedral edge that guarantees an accurate approximation of all of the materials can be found:

$$a \leq \frac{2\arcsin\left(\sqrt{\epsilon_m \kappa_{m_i,max}(2-\epsilon_m \kappa_{m_i,max})}\right)}{\kappa_{m_i,max}} \tag{6.27}$$

where $\kappa_{m_i,max} = \max\left\{\kappa_{m_0,max}, \kappa_{m_1,max}, \ldots \kappa_{m_{d_m-1},max}\right\}$. If the region of smoothly graded composition is approximated by a tetrahedral mesh of the above dimension, the error in approximation for each material is guaranteed to be less than ϵ_m.

Substituting the expression for a as a function of material curvature back into Equations 6.19 and 6.20, the volume and number of tetrahedra as a function of maximum material curvature is

found.

$$
\begin{aligned}
n_{tetrahedra} &= O\left\lceil \frac{6\sqrt{2}V_{region}}{a^3} \right\rceil \\
&= O\left\lceil 6\sqrt{2}\left\lceil \frac{\kappa_{m_i,max}}{2\arcsin\left(\sqrt{\epsilon_m \kappa_{m_i,max}(2-\epsilon_m \kappa_{m_i,max})}\right)} \right\rceil \right\rceil
\end{aligned} \tag{6.28}
$$

6.5 Relationship between the number of triangles and nodes in a triangulated shell

The preceding sections derived relationships between the number of triangles in a mesh as a function of the desired shape of an object. In triangulated B-rep data structure, however, the storage costs are not just a function of how many triangles are present, but also a function of the number of nodes (unique vertices) required to represent the mesh. Although the relationship between the number of nodes and triangles is completely dependent on the topology of the model (or how the triangles are related to each other), a bound for the number of nodes as a function of the number of triangles can be determined.

To begin this analysis, consider a set of three operations for creating and modifying a triangulated mesh.

Op_I: Remove one triangle. Insert one node, three edges, and three triangles.

Op_{II}: Remove two triangles and an edge. Insert one node, four edges, and four triangles.

Op_{III}: Remove two triangles to create a hole. Add six edges and six triangles to form the walls of the hole.

Starting with a tetrahedron, any closed, triangulated mesh can be created by repeatedly applying the above operations. The first two modify the mesh by adding nodes, to create a net increase of two new triangles in the mesh, as illustrated in Figures 6-11(a) and (b). The third method does not change the number of nodes in the mesh, but increases the number of triangles to form the boundary of a hole through the region, as in Figure 6-11(c). Table 6.1 summarizes how the number of features in the mesh are changed after the application of each operation. To relate the number of nodes in

	Operation		
Feature	Op_I	Op_{II}	Op_{III}
$n_{triangles}$	+2	+2	+4
n_{nodes}	+1	+ 1	
n_{holes}			+1

Table 6.1: Modification to triangulated mesh.

the mesh to the number of triangles, let $n_{newnodes}$ represent the number of times operations I or

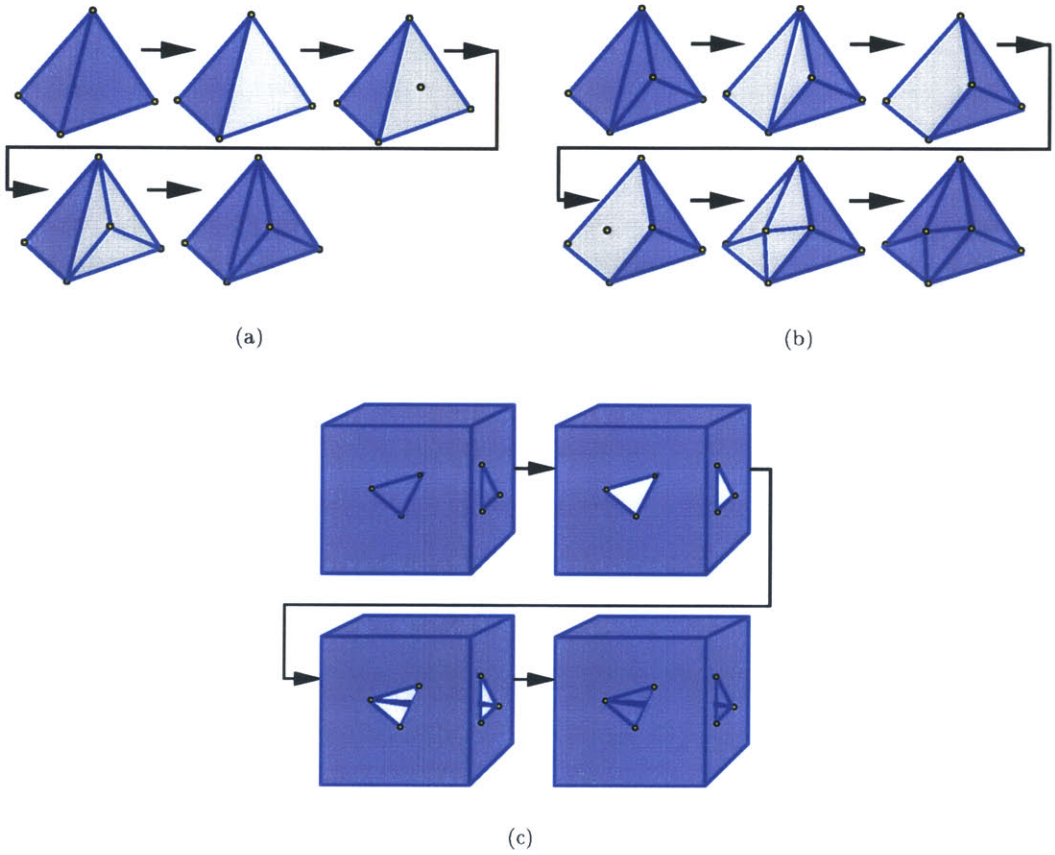


Figure 6-11: Topological operations for modifying a closed, triangulated shell. (a) The subdivision of a face into three triangles (Op_I). (b) The subdivision of an edge into two edges (Op_{II}). (c) The creation of a hole through the shell (Op_{III}).

II are applied and let n_{holes} represent the number of holes (Op_{III}) added to the model. From the initial model of a tetrahedron, the number of features after a sequence of operations is:

$$n_{triangles} = 4 + 2n_{newnodes} + 4n_{holes} \text{ and } n_{nodes} = 4 + n_{newnodes}. \tag{6.29}$$

Therefore, the total number of nodes in a triangulated mesh model as a function of the number of triangles and holes in the model is:

$$\begin{aligned} n_{nodes} &= n_{nodes}[n_{triangles}, n_{holes}] \\ &= \frac{1}{2}n_{triangles} - 2n_{holes} + 2 \end{aligned} \tag{6.30}$$

6.6 Relationship between the number of tetrahedra and nodes in a finite element mesh

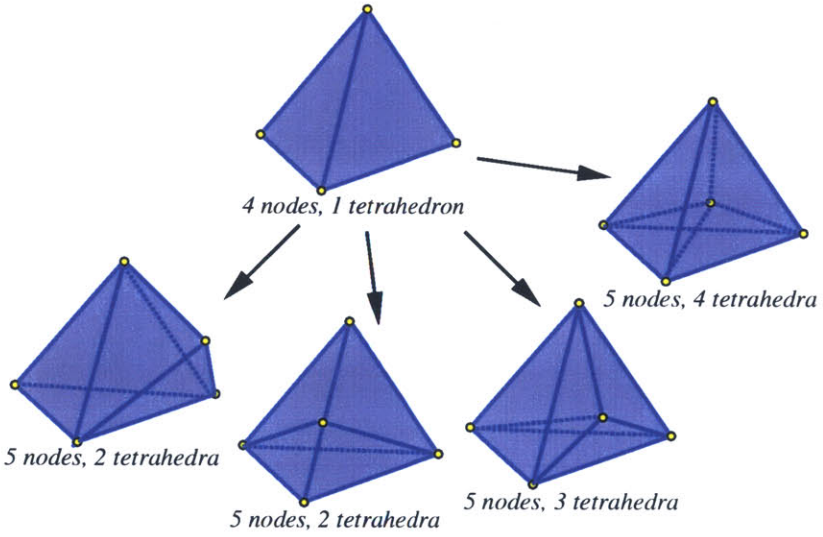


Figure 6-12: Methods to modify a tetrahedron with the addition of a new node.

As in the preceding section concerning models of triangulated shells, the relationship between the number of tetrahedra and nodes in a volumetric mesh is entirely dependent on the topology of the model, or how the mesh is constructed. Unlike the analysis of the triangulated mesh in which there were only three operations for modifying its topology, there are many ways to modify the topology of a tetrahedral mesh. Only the most extreme case resulting in an upper bound for the number of nodes as a function of the number of tetrahedra is explored here.

Consider the case in which a tetrahedral mesh model is generated from an initial tetrahedron. Each node added to the model is added outside the mesh, resulting in one new tetrahedron, three

new faces, and three new edges.

$$n_{tetrahedra} = 1 + n_{newnodes} \tag{6.31}$$

Inverting this, the ratio of nodes added to the model per tetrahedron is $1:1$. This represents the worst case in terms of memory: the maximum number of nodes per tetrahedra. All other possible methods of adding a node (subdividing an existing tetrahedron, triangle, or edge) result in multiple tetrahedra. Therefore, the upper bound for the number of nodes in a tetrahedral mesh model is bounded by:

$$n_{nodes} = O\left(n_{tetrahedra} + 3\right). \tag{6.32}$$

6.7 Discussion

This chapter has introduced issues concerning the growth of meshed models (triangulated shells and tetrahedral meshes). In each case, the number of elements in the meshes are a function of the maximum edge size that still guarantees the geometric and material accuracy. Although adaptive meshing techniques could be used to reduce the number of elements, only uniform meshes are considered for simplicity of analysis, illustrating the trends in growth of meshed models as functions of accuracy.

The triangulated shells, the maximum size of the triangles guaranteeing the accurate approximation of the boundary is found to be a function of:

1. the minimum geometric feature size (μ_g) in a bounding curve or surface, and
2. the maximum geometric curvature ($\kappa_{g,max}$) of a bounding curve or surfaces.

The above characteristics concerning the shape of the model determine the maximum dimension of an edge in a triangle that still guarantees the geometric accuracy. Assuming a uniform mesh of equilateral triangles of this dimension, the number of triangles required to cover the surface is readily found. The number of nodes in a triangulated mesh, however, is dependent on the topology (genus) of the object (number of holes). For a given number of triangles, however, an upper bound on the number nodes can be established. Substituting this information for the growth of triangulated shells into the expression for the storage cost from the preceding chapter (Equation 4.5) , an upper bound on the storage cost of triangulated shells as a function of geometric accuracy (ϵ) as well as

the intended shape of the model (μ_g and κ_g) is formed:

$$\begin{aligned}
S_{tri} &= S_{tri}[n_{regions}, n_{triangles}, n_{nodes}] \\
&= (S_{int} + S_{ptr} + d_m S_{flt}) n_{regions} + (5S_{ptr} + 2S_{bln}) n_{triangles} + 3S_{flt} n_{nodes} + \\
&\quad S_{int} + S_{ms} \\
S_{tri} &= S_{tri}[n_{regions}, n_{holes}, A_{surface}, \epsilon_g, \kappa_{g,max}, \mu_g] \leq S_{tri}[n_{regions}, n_{triangles}[A_{surface}, \epsilon_g, \kappa_{g,max}, \mu_g]] \\
&\leq (S_{int} + S_{ptr} + d_m S_{flt}) n_{regions} + S_{int} + S_{ms} + \\
&\quad (5S_{ptr} + 2S_{bln} + \frac{3}{2} S_{flt}) n_{triangles}[A_{surface}, \epsilon_g, \kappa_{g,max}, \mu_g] \qquad (6.33)
\end{aligned}$$

where $n_{triangles}$ is determined by the surface area of the model, the minimum geometric feature size (μ_g) in any surface or bounding curve in the model, by the maximum of the curvature ($\kappa_{g,max}$) of any surface or bounding curve in the model, as well as the prescribed geometric accuracy, according to Equation 6.18:

$$\begin{aligned}
n_{triangles}[A_{surface}, \epsilon_g, \kappa_{g,max}, \mu_g] &= O\left[\frac{4\sqrt{3} A_{surface}}{3a^2} \right] \qquad (6.34) \\
\text{where } a &= \min\left\{ \mu_g, \frac{\sqrt{3}}{\kappa_{g,max}} \left(\arcsin \sqrt{\epsilon_g \kappa_{g,max} (1 - \epsilon_g \kappa_{g,max})} \right) \right\}
\end{aligned}$$

Tetrahedral models are subject to the same constraints imposed by the minimum geometric features size and maximum geometric curvature. In addition, the nature of the intended material variation ($\mathbf{m}^*(\mathbf{x})$) may further restrict the size of the tetrahedra. This chapter identifies two of these factors:

1. The minimum intended material feature size within a region (μ_m) or
2. The maximum intended material curvature within a region ($\kappa_{m,max}$).

Along with the previous geometric constraints for triangulated shells, the maximum edge length of the tetrahedra that guarantees the accurate approximation of the intended geometry and composition can be formed. From this, a bound on the number of tetrahedra is found.

The number of tetrahedra, however, is only one factor affecting the total storage cost of a meshed model. The other is the number of nodes in the mesh. Although the relationship between the nodes and tetrahedra is entirely dependent on the topology of the model (how the tetrahedra are connected together), an upper bound for the worst case topology can be established, as presented in Equation 6.32.

From these relationships, the growth of tetrahedral models as a function of geometric and material accuracys (ϵ_g and ϵ_m) as well as the intended shape and material grading ($V_{interior}$, $A_{surface}$, μ_g,

μ_{mt}, κ_g, and κ_m) can be formed from Equations 4.11 through 4.14:

$$
\begin{aligned}
S_{te_{xx}} &= S_{te_{xx}}[n_{tetrahedra}, n_{nodes}] \\
&\leq S_{te_{xx}}[V_{interior}, \epsilon_g, \epsilon_m, \mu_g, \mu_{mt}, \kappa_g, \kappa_m] && (6.35) \\
S_{te_{00}} &\leq [4S_{ptr} + (3+d_m)S_{flt}]n_{tetrahedra} + && (6.36) \\
&\qquad 3(3+d_m)S_{flt} + S_{int} + S_{ms} + S_{ptr} \\
S_{te_{01}} &\leq [4S_{ptr} + (4d_m+3)S_{flt}]n_{tetrahedra} + && (6.37) \\
&\qquad 9S_{flt} + S_{int} + S_{ms} + S_{ptr} \\
S_{te_{10}} &\leq [8S_{ptr} + (3+d_m)S_{flt} + S_{int}]n_{tetrahedra} + && (6.38) \\
&\qquad 3(3+d_m)S_{flt} + 4S_{int} + S_{ms} + S_{ptr} \\
S_{te_{11}} &\leq [8S_{ptr} + (4d_m+3)S_{flt} + S_{int}]n_{tetrahedra} + && (6.39) \\
&\qquad 9S_{flt} + 4S_{int} + S_{ms} + S_{ptr} && (6.40)
\end{aligned}
$$

where $n_{tetrahedra}$ is determined by the volume of the model, the minimum geometric (μ_g) feature of any surface patch or surface patch's bounary curve, minimum material (μ_{mt}) feature size, the maximum geometric curvature($\kappa_{g,max}$) of any surface patch or surface patch's boundary curve, or material (κ_m) curvature in the intended model, as well as the prescribed accuraciess. Summarizing these relationships from this chapter, the number of tetrahedra as a function of these variables is:

$$
n_{tetrahedra}[V_{interior}, \epsilon_g, \mu_g, \kappa_{g,max}, \epsilon_m, \mu_{mt}, \kappa_{m,max}] = O\left[\frac{6\sqrt{2}V_{interior}}{a^3}\right]
$$

$$
\begin{aligned}
\text{where } a &= \min\left\{\frac{\sqrt{3}}{\kappa_{g,max}}\arcsin\sqrt{\epsilon_g\kappa_{g,max}(1-\epsilon_g\kappa_{g,max})},\ \mu_g,\right. \\
&\qquad \left.\frac{2}{\kappa_{m,max}}\arcsin\sqrt{\epsilon_m\kappa_{m_i,max}(2-\epsilon_m\kappa_{m,max})},\ \mu_{mt}\right\}.
\end{aligned}
$$

Note that for tetrahedral meshing, the minimum feature size (μ_{mt}) is not only measured between material boundaries, but between discontinuities in the material derivatives or between a discontinuity in the material derivative and a material boundary.

In conclusion, this chapter has identified and quantified issues concerning the approximation of an intended FGM design with either triangular facets or linear tetrahedra. For simplicity in analysis, the restriction that the meshes consist of uniformly sized elements was imposed. Although models could be meshed with fewer elements than predicted here, the results found here serve as an upper bound for the maximum number of elements in each case and predicts the rate of growth as a functions of geometric and material accuracys. Geometric feature size and curvature, factors considered here, are regularly considered in commericial meshing systems during the approximation

of solid models with finite elements. For the establishment of a solid modeling representation based solely on finite elements, meshing methods will have to be extended, taking into account the intended material feature size and curvature, both of which were described in this chapter and play a role in limiting the distance between nodes in the tetrahedral mesh.

Chapter 7

Approaches to FGM design

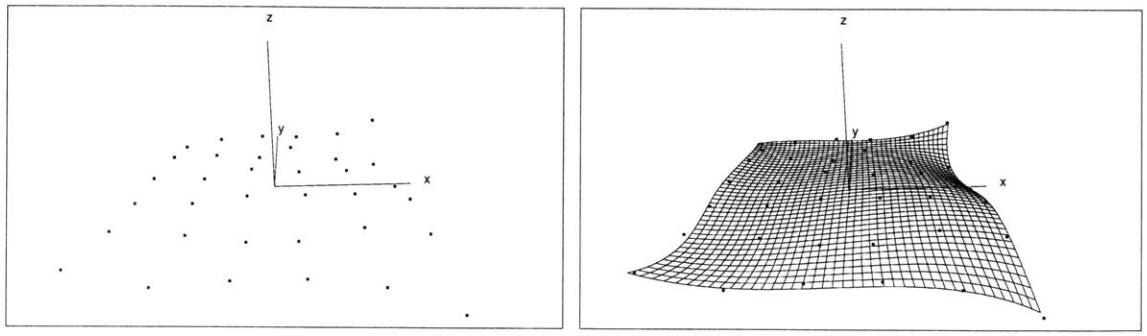
7.1 Motivation

The concept of designing Functionally Graded Material (FGM) objects is new to designers and manufacturers. Existing CAD systems allow at most users to create piecewise constant FGM objects using composite structures and assemblies by associating *attributes* with solids. Methods to capture the designer's intent for graded compositions have yet to be defined. The development of methods for FGM design, however, directly impact the choice of data structure for FGM solid modeling since these tools determine the range of parts that can be defined. In order to understand how the designer's intent might be captured, this chapter proposes three different classes of design tools and how they might be used for FGM object creation. These tools are then used as the pathways for capturing design intent in subsequent chapters, allowing the storage costs associated with the various data structure to be related back to the designer's original intent. The three classes of design tools proposed in this chapter include: FGM fitting, design from an FGM library, and FGM blending.

7.2 FGM fitting

The first class of FGM design tools fall into the category of "fitting". In this category, a desired material variation is somehow defined and then approximated by the methods available for representing graded compositions within the data structure. This analogous to geometric surface design by specifying a set of points and then approximating or interpolating the data with a surface (see Figure 7-1).

An example of FGM fitting to some target function $\mathbf{m}^*(\mathbf{x})$ is the design of graded compositions as functions of distance within a finite element data structure [32]. Field values for the nodes in the mesh are assigned according to their position: $\mathbf{m_i} \leftarrow \mathbf{m}^*(\mathbf{x_i})$ where $\mathbf{m_i}$ is the vector of volume



(a) (b)

Figure 7-1: (a) Set of data points. (b) Surface fit of data points.

fractions associated with node $\mathbf{i}$ at the point $\mathbf{x_i}$. In general, the modeled function will not exactly represent the intended grading except at the evaluated nodes. A target function may be cubic, for instance, but a mesh consisting of linear elements will only provide an approximation of the desired function.

One example of FGM fitting, that has been proposed for the design of compositions involves a function of distance. In this approach, some reference feature is defined, such as a point, line (eg., axis), plane, or collection of triangles, and the composition is designed as a function of distance from that feature. The desired variation may be any function of the distance (r) from the feature. In the current implementation of the design system based on a finite element mesh representation, the design function may be described in terms of a start (r_s) and an ending distance (r_e), the material volume fractions at these distances, and the desired grading over these distances. The nodes that fall within the specified range are assigned the desired composition: $\mathbf{m_i} \leftarrow \mathbf{m}^* [r(\mathbf{x_i})]$ if $r_s \leq r(\mathbf{x_i}) \leq r_e$. Figure 7-2 illustrates the area over which a composition may be defined as a function of distance in two dimensions, with one node lying within the range of assignment and two lying outside. The current implementation allows the assignment of a uniform composition within the desired range, linear variation, quadratic variation (with smooth blending at r_s or r_e), or cubic grading (with smooth blending at both r_s and r_e). These methods of grading are illustrated in Figures 7-3(a)-(e) in which the weights given to the two compositions are plotted as a function of parameter t such that:

$$t = t(\mathbf{x_i}) = \frac{r(\mathbf{x_i}) - r_s}{r_e - r_s} \tag{7.1}$$

$$\mathbf{m_i} = \mathrm{w}_s(t)\mathbf{m}_s^* + \mathrm{w}_e(t)\mathbf{m}_e^* \tag{7.2}$$

The compositions $\mathbf{m}_s^*$ and $\mathbf{m}_e^*$ are the intended vectors of volume fractions to be assigned at the distances r_s and r_e from the feature, respectively.

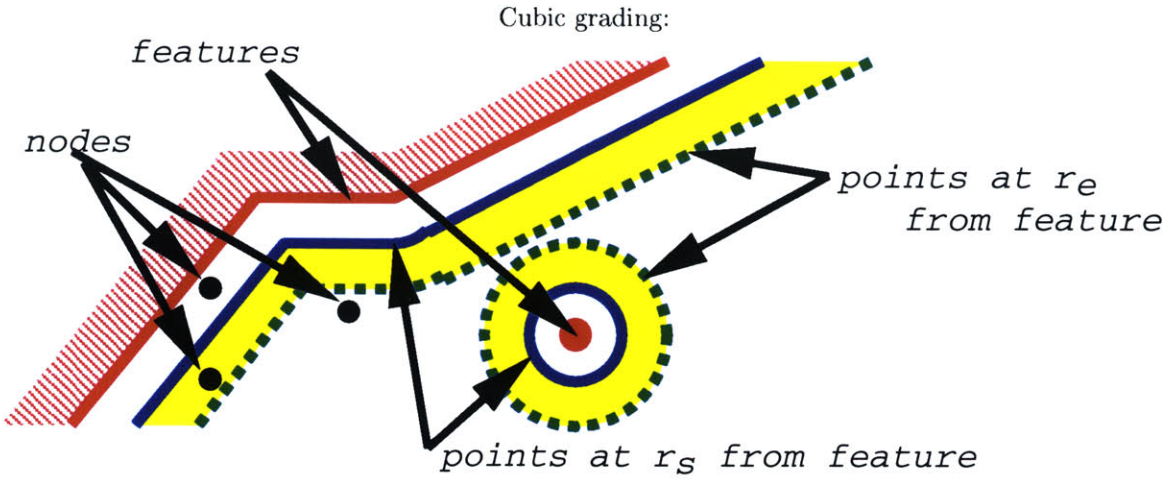


Figure 7-2: Illustration of evaluation of distance from features (red) to node points (black) and the offset region (yellow) in which the composition is to be fitted.

Figures 7-4- 7-10 illustrate how the composition over a finite element mesh may be assigned by fitting the field values at the nodes of the mesh to a target function.

Figures 7-4 and 7-5 demonstrate the specification of a composition designed from a point within a cube of linear tetrahedra. The Material System in this case consists of three Materials. All of the nodes that are within a distance of $\frac{1}{2}$cm of the point (0,0,0) are first assigned a composition according to a quadratic function grading between two compositions, as shown in Figure 7-4. Next, compositions are assigned to the nodes that are at least $\frac{1}{2}$cm distant from point (0,0,0) but no further than 1 cm, as illustrated in Figure 7-5. The end result is a mesh whose nodes approximate a piece-wise, quadratic material variation, with composition $[0\ 1\ 0]^T$ at point $(0,0,0)$, composition $[1\ 0\ 0]^T$ at a distance of $\frac{1}{2}$cm from point (0,0,0), and composition $[0\ 0\ 1]^T$] beyond a distance of 1cm from point (0,0,0).

The next example, in Figure 7-6, illustrates the design of composition from the diagonal of the cube, from point (0,0,0) to (1,1,1). Compositions are assigned to the nodes that lie within a distance of $\frac{2}{3}$cm of the line, resulting in an approximation of a target function by the tetrahedra stored in the mesh. In this case, the target function is a cubic variation, with the composition $\mathbf{m}_s = [1\ 0]^T$ along the line and composition $\mathbf{m}_e = [0\ 1]^T$ over the surface of a cylinder with radius $\frac{2}{3}$ cm aligned with its axis along the diagonal of the cube.

Figure 7-7 illustrates the design of a composition from a plane. In this case the target variation is linear in nature, grading from $m_s = [1\ 0]^T$ at the plane $\pi : z = 0cm$ to $m_e = [0\ 1]^T$ at the plane $\pi : z = 1cm$.

Figures 7-8 and 7-9 illustrate the fit of a composition to a quadratic function of distance from the boundary. Figure 7-8 is the initial model, represented within a commercial CAD system. The model is then meshed into near uniform tetrahedral elements, generating the nodes shown in Figure 7-9.

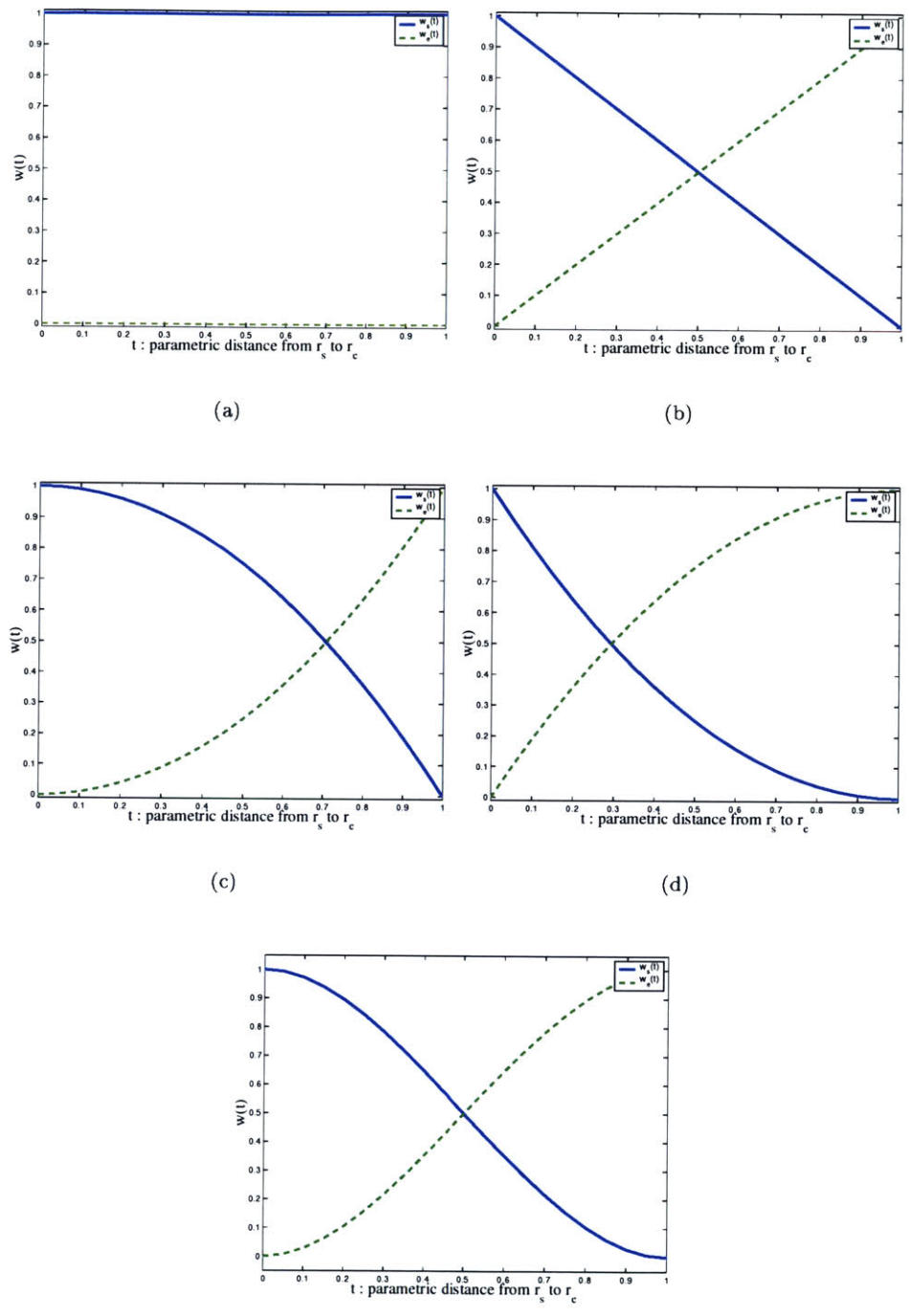
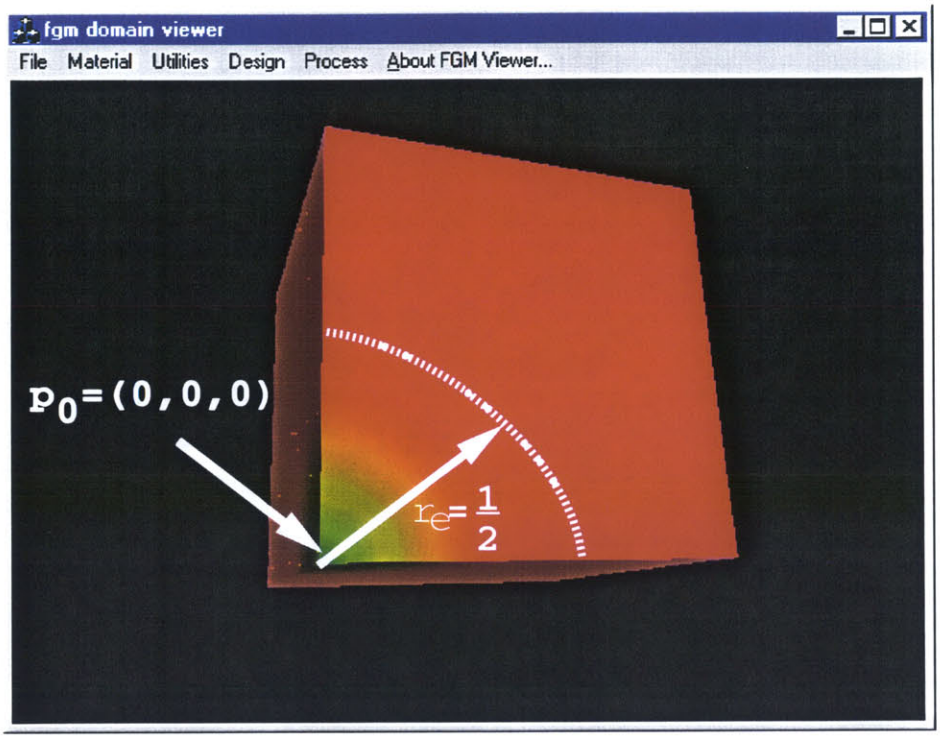


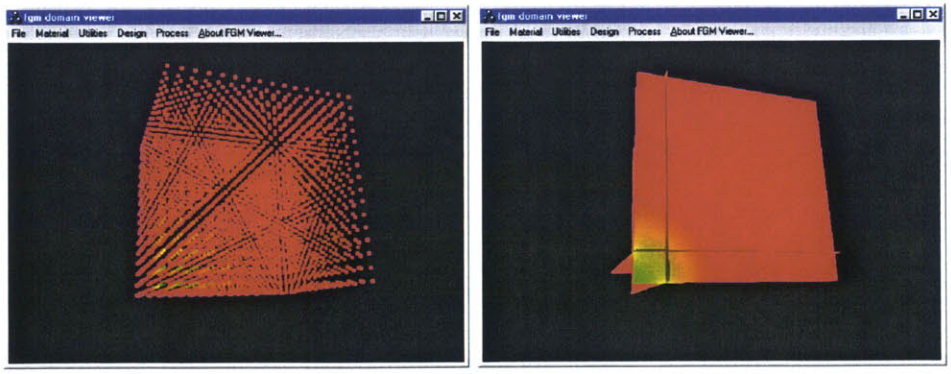
Figure 7-3: Grading styles for the design of FGM objects as a function of distance: (a) uniform, (b) linear, (c) and (d) quadratic, and (e) cubic.

In this case, a desired composition of $\mathbf{m}_s = [0\ 1]^T$ is assign to the boundary of the model which grades quadratically over a distance of 1mm to $\mathbf{m}_e = [1\ 0]^T$ within the model's interior. The minimum distance of each node to the boundary of the model is computed and its composition defined according to the prescribed function. Figure 7-9(a) shows the nodes colored according to their assigned compositions. Slices through the model material variation are illustrated in Figure 7-9(b). One can see in this illustration how the idealized, quadratic function has been approximated by the piece-wise linear tetrahedra, resulting in oscillations over the nearest slicing plane.

The final example, Figure 7-10 illustrates the fit of a composition designed from a subset of the model's boundary. In this case, the composition is designed as a quadratic function of distance from the cavity of the model, with the desired grading occurring over a distance of 5mm.

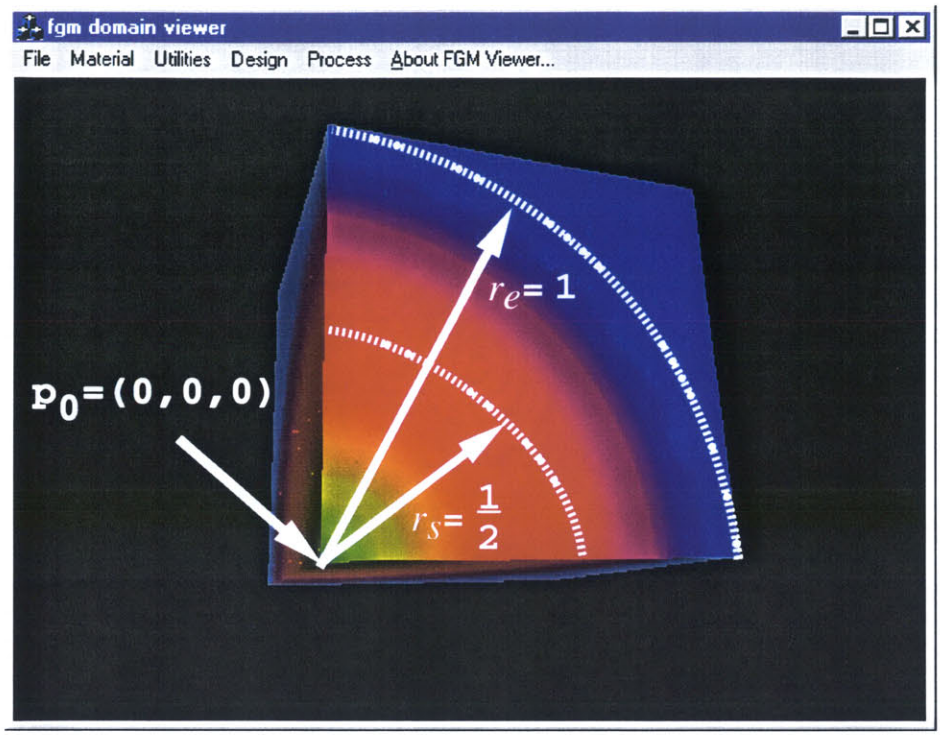


(a)

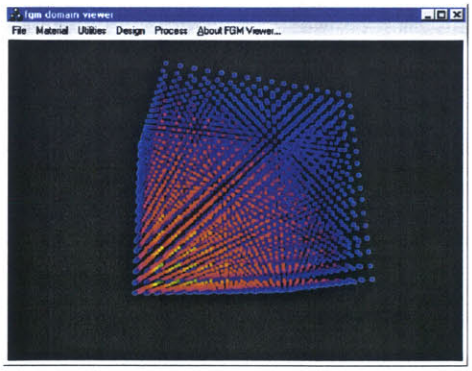


(b) (c)

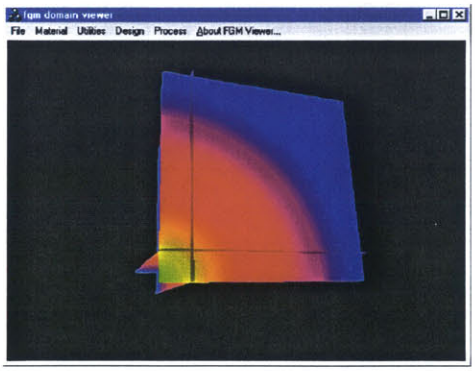
Figure 7-4: (a) Fit of composition (smoothly blended at r_e) designed as a quadratic function of distance within a unit cube from point $\mathbf{p}_0 = (0,0,0)$ ($r_s = 0$cm, $r_e = \frac{1}{2}$cm, $\mathbf{m}_s = [0\ 1\ 0]^T$, $\mathbf{m}_e = [1\ 0\ 0]^T$). (b) Rendering of nodes colored according to composition. (c) Rendering of material variation over slices through cube.



(a)

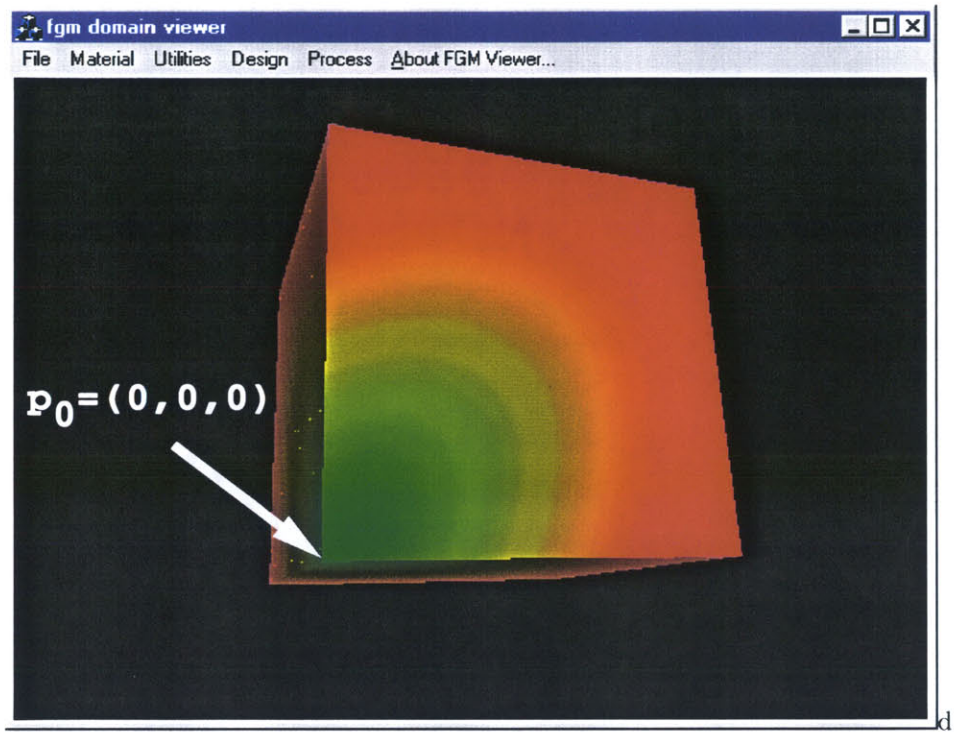


(b)

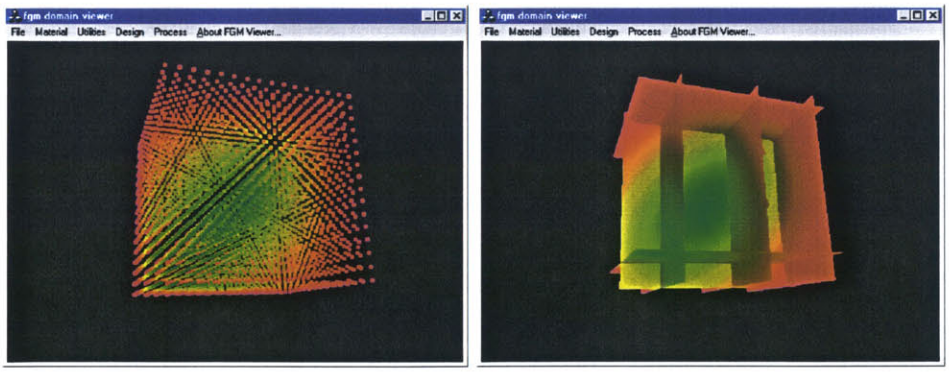


(c)

Figure 7-5: (a) Previous design plus fit of composition (smoothly blended at $r_s = \frac{1}{2}$cm and $r_e = 1$cm) designed as a cubic function of distance within a block from point $\mathbf{p}_0 = (0,0,0)$ ($\mathbf{m}_s = [1\ 0\ 0]^T$, $\mathbf{m}_e = [0\ 0\ 1]^T$). A uniform composition of $\mathbf{m} = [0\ 0\ 1]^T$ is assigned to all nodes beyond a distance of 1 mm from point $\mathbf{p}_0 = (0,0,0)$. (b) Rendering of nodes colored according to composition. (c) Rendering of material variation over slices through cube.

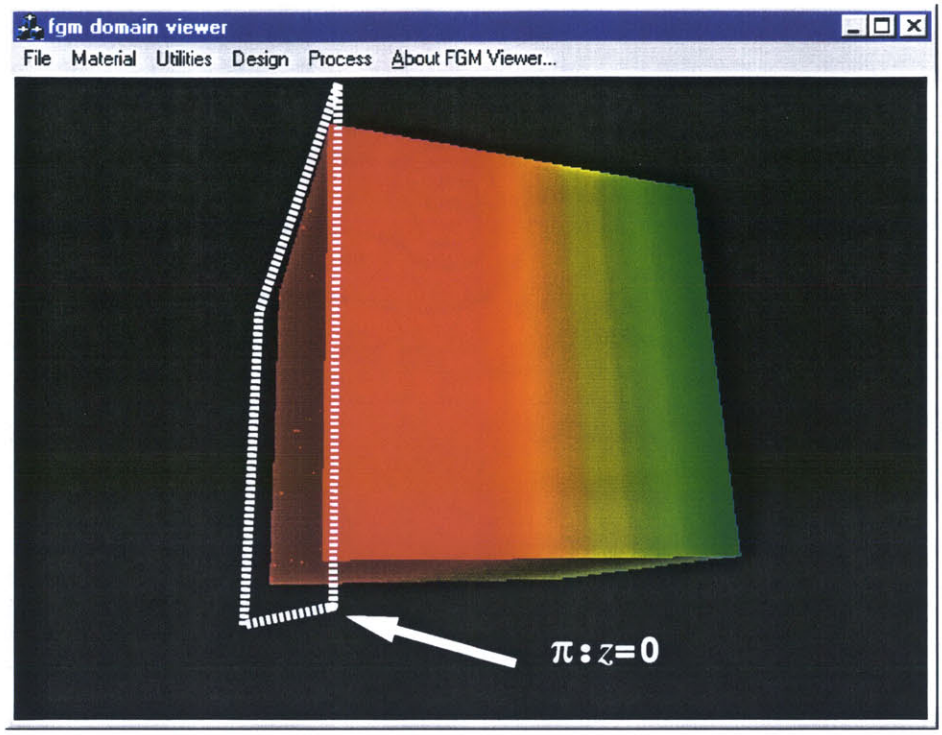


(a)

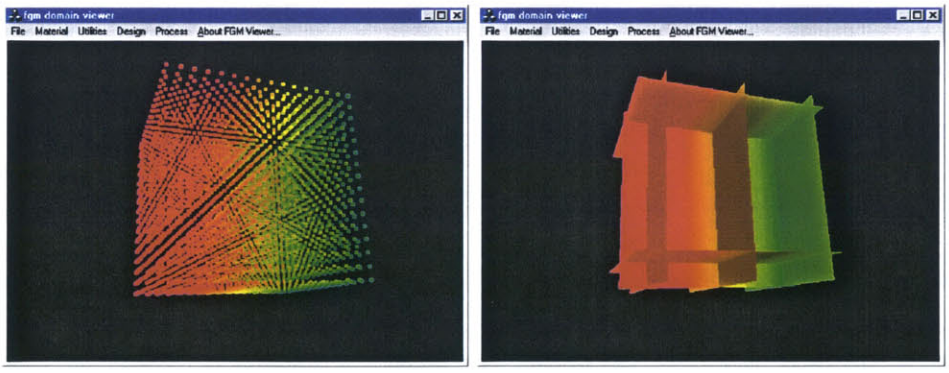


(b) (c)

Figure 7-6: (a) View of solid cube with composition designed as a cubic function of distance within a unit cube as from the line passing through $\mathbf{p}_0 = (0,0,0)$ to $\mathbf{p}_1 = (1,1,1)$ ($r_s = 0$cm, $r_e = \frac{2}{3}$cm, $\mathbf{m}_s = [1\ 0]^T$, $\mathbf{m}_e = [0\ 1]^T$). (b) View of nodes in mesh. (c) View of slices through FGM cube.



(a)



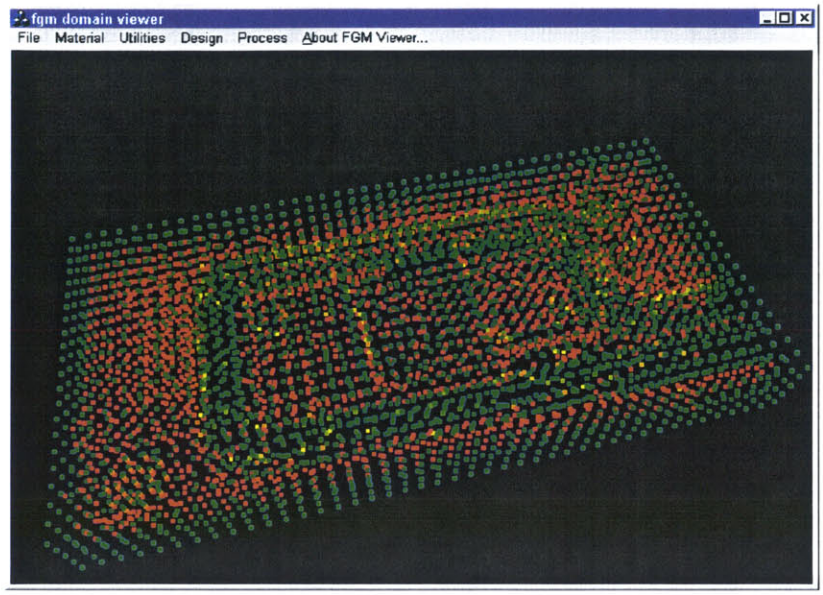
(b) (c)

Figure 7-7: Fit of composition designed as a linear function of distance within a unit cube from plane $\pi : z = 0$ ($r_s = 0$cm, $r_e = 1$cm, $\mathbf{m}_s = [1\ 0]^T$ and $\mathbf{m}_e = [0\ 1]^T$. (a) View of solid cube. (b) View of nodes in mesh. (c) View of slices through FGM cube.

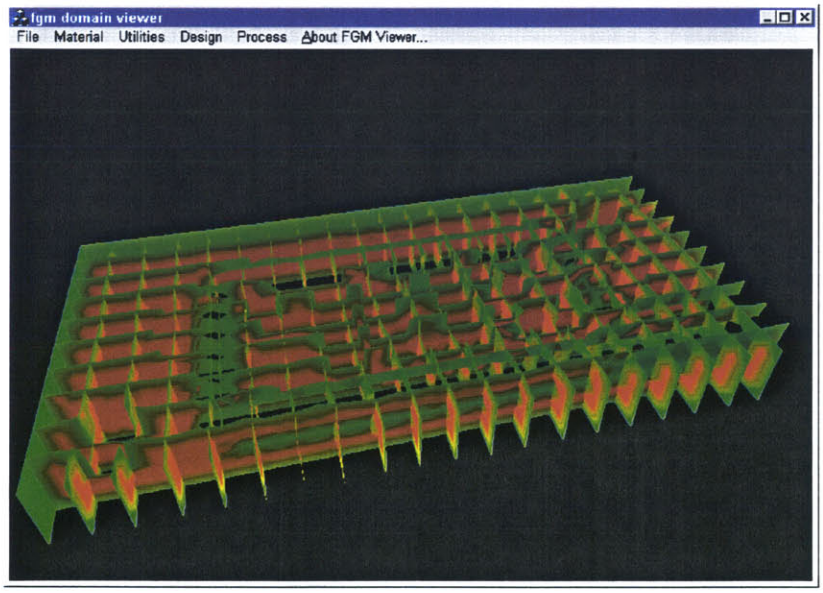
(a)

(b)

Figure 7-8: (a) Design of tool on commerical CAD system. The dimension of the tools is $100mm \times 50mm \times 10mm$. (b) Phantom view of the tool showing internal features (cooling channels).

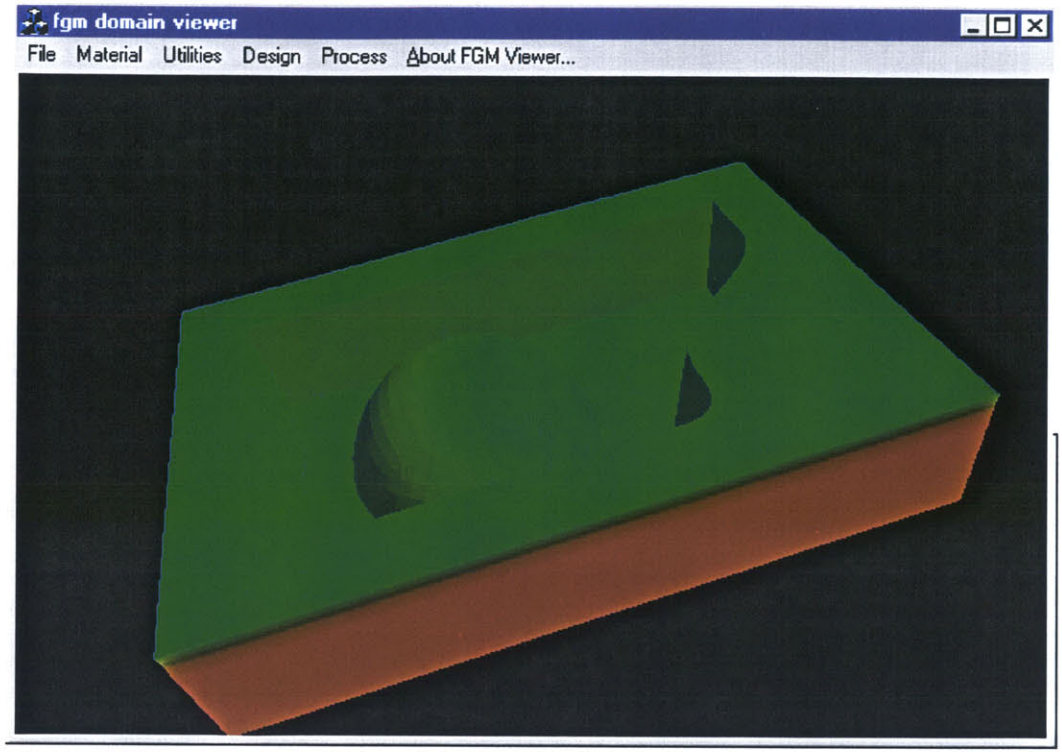


(a)

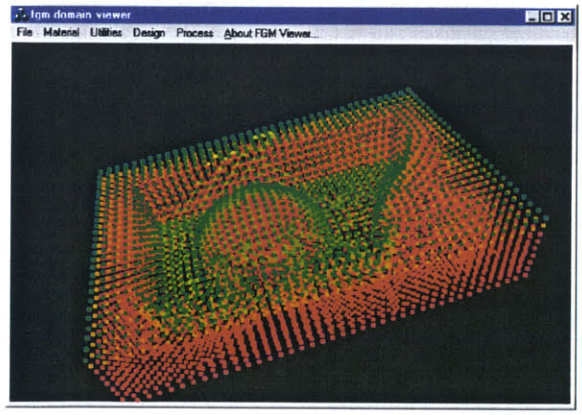


(b)

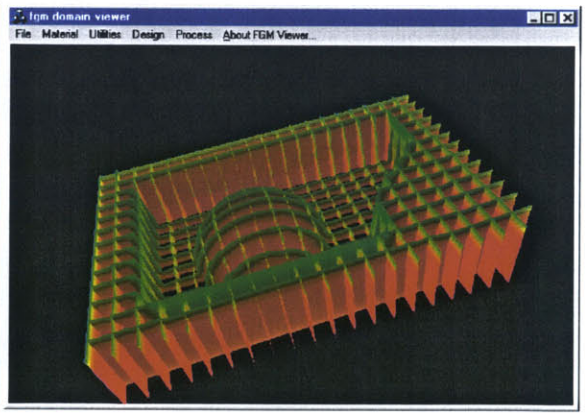
Figure 7-9: (a) Fit of nodes to intended composition (smoothly blended at r_e) designed as a quadratic function of distance from the boundary of the tool ($r_s = 0\text{mm}$, $r_e = 1\text{mm}$). (b) View of composition grading over slices through FGM model.



(a)



(b)



(c)

Figure 7-10: Fit of composition (smoothly blended at r_e) designed as a quadratic function of distance from a subset of the boundary of a tool ($r_s = 0$mm, $r_e = 5$mm). The dimension of the tool is 100mm $\times 66\frac{2}{3}$mm $\times 20$ mm. (a) View of solid tool. (b) View of nodes in mesh. (c) View of slices through FGM model.

7.3 FGM library

Another approach to designing FGM objects is through the use of a library of FGM primitives. This approach to FGM design is an extension of the use of libraries in solid modeling for shape design or logic libraries in VLSI design.

In commercial CAD systems, the repeated design of features on a part can be simplified and automated through the use of a library. The feature is first designed and stored in a library. As a new part is created and a feature must added, instead of recreating the entire feature for each instance, it is simply retrieved from the library and then mapped over the part. Figure 7-11(a) is an example of a feature that might be stored in a library as a primitive for texturing surfaces [13, 36]. The application of the feature over two different parts is illustrated in Figures 7-11(b) and (c).

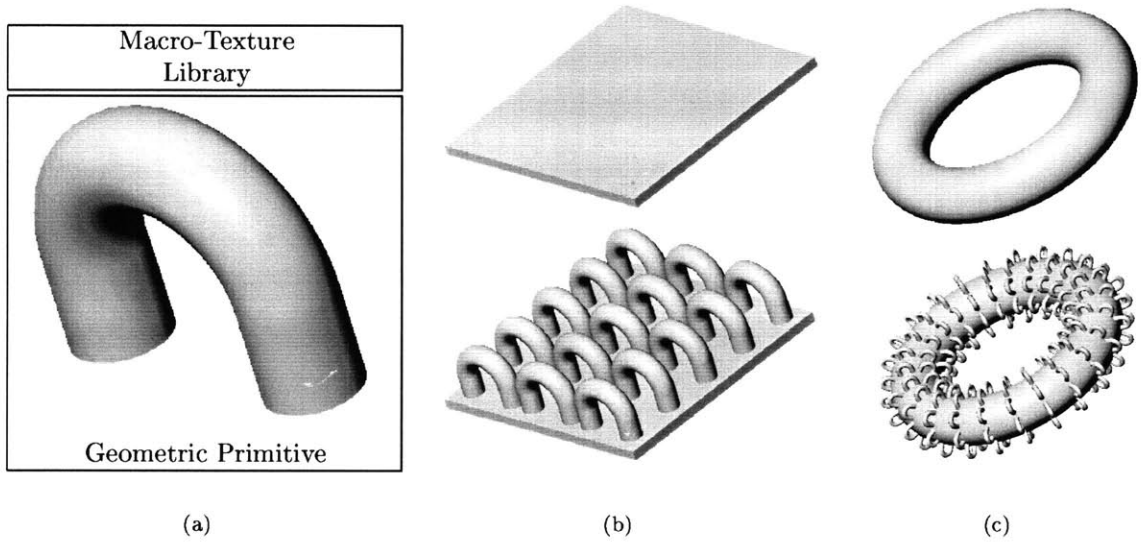


Figure 7-11: (a) A texture primitive stored in a feature library. (b) The mapping of the feature over a plate. (c) The mapping of the feature over a torus.

The same approach to the design from a library could be applied to the material design of FGM objects. A library of material primitives (designed to perform some specific task) could be defined and then mapped into a part. With each primitive's addition to the model, the primitive's material variation overwrites the previously designed composition in the part such that:

$$\mathbf{m}^*_{final}(\mathbf{x}) = \begin{cases} \mathbf{m}^*_{primitive}(\mathbf{x}) \text{ for } \mathbf{x} \in \text{ FGM primitive} \\ \mathbf{m}^*_{initial}(\mathbf{x}) \text{ otherwise} \end{cases} \tag{7.3}$$

One example of how FGM primitives might be used to design an object is given in Figure 7-12. In this example, a drug delivery device is created by mapping FGM primitives from a library into a pill. The position of each primitive of the pill relative to the rate at which the pill is absorb determines

when the drug in the primitive would be released [78]. This allows drug designers to tailor the release profile for optimal delivery.

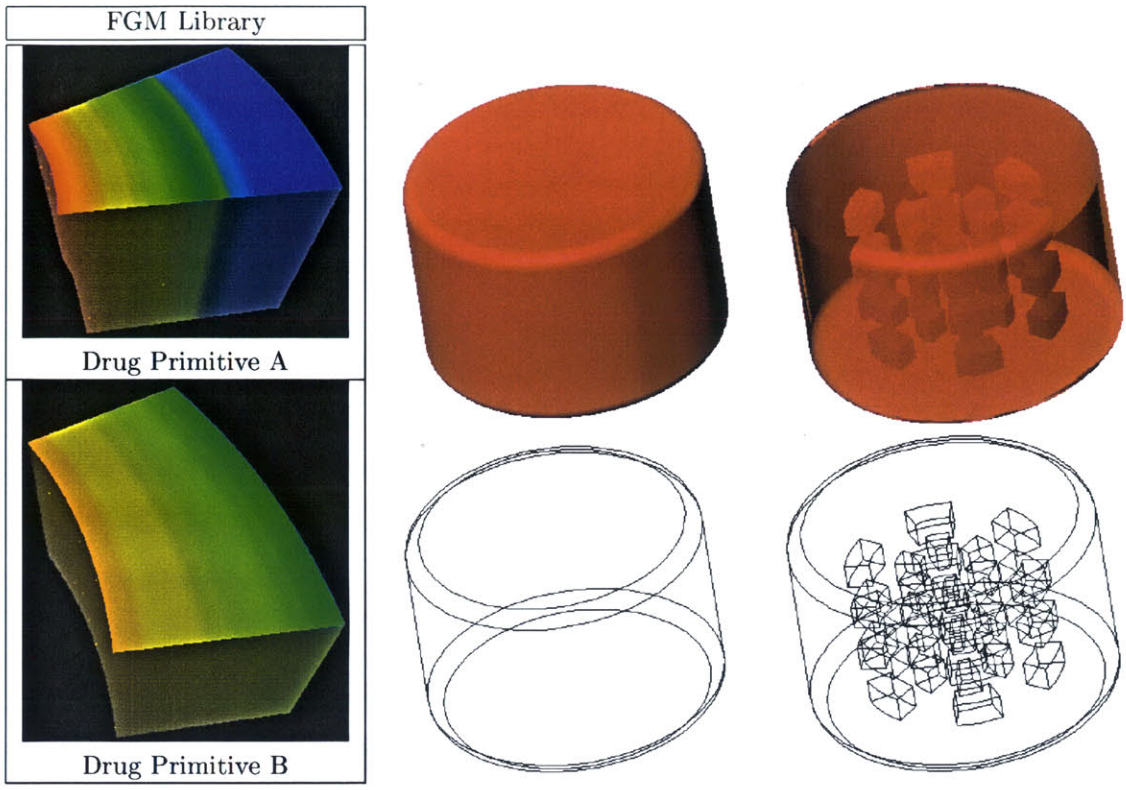
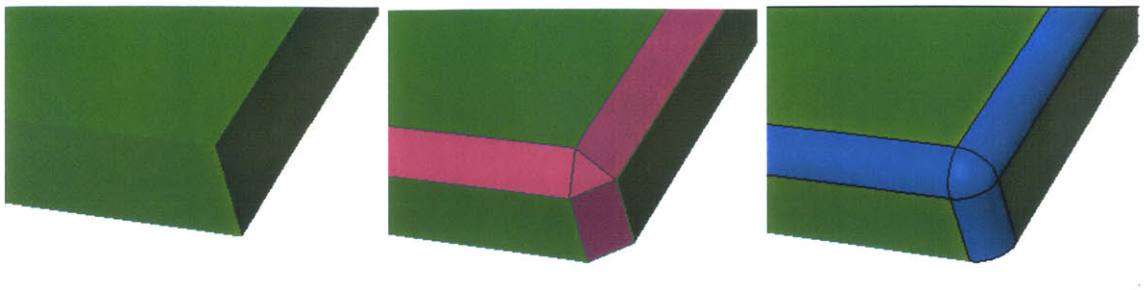


Figure 7-12: (a) Primitive of regions containing drug. (b) The initial pill consisting of a uniform base material. (c) The mapping of drug primitives into the drug delivery device. The placement of of the primitives tailors the drug release profile.

7.4 FGM chamfer, fillet, and blending

Chamfer, fillet, and blend operations are commonly used in Computer Aided Geometric Design to define the shape of a part (see Figure 7-13). The same approach may be taken for the design of FGM objects. Just as chamfer, fillet, and blend operations create a new surface to join two tangent discontinuous surfaces, material chamfer, fillet, and blend operations could be used to generate intermediate regions to join two differing regions.

For a material chamfer operation, a face is selected and an offset distance Δ specified. A region of uniform composition would then be generated, bounded by new faces offset distance Δ into the neighboring regions from the original face. The composition assigned to the interface region could



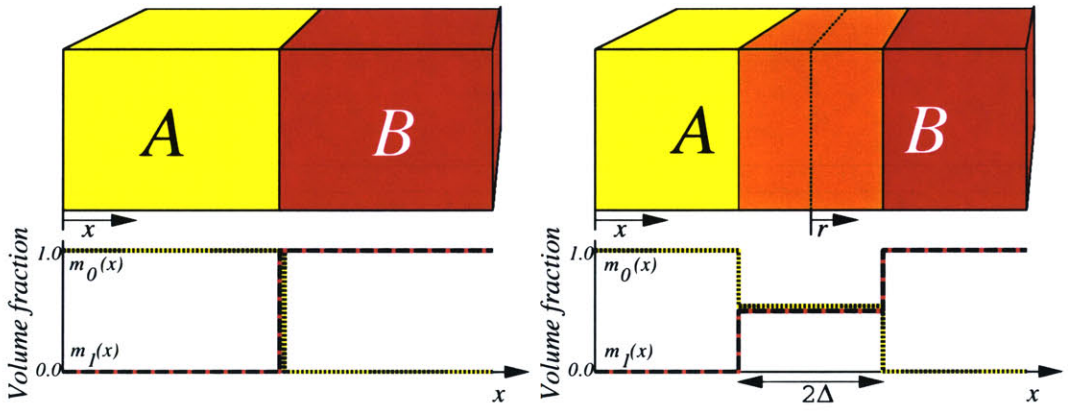
(a) (b) (c)

Figure 7-13: (a) Original design of corner of part. (b) Chamfered corner. (c) Filleted corner.

be specified or taken as the average of the two neighboring regions' compositions:

$$\mathbf{m}^*_{final}(\mathbf{x}) = \begin{cases} \frac{1}{2}\left(\mathbf{m}^B + \mathbf{m}^A\right) & \text{for } \mathbf{x} \in \text{ chamfer region} \\ \mathbf{m}^*_{initial}(\mathbf{x}) & \text{otherwise} \end{cases} \tag{7.4}$$

Figure 7-14 illustrates the result of this operation in 2D.



(a) (b)

Figure 7-14: (a) Original material distribution over a cross-section of a block. (b) Material distribution over block cross-section with material chamfer.

The material fillet operation could be performed two ways. The first method would require the selection of a face and specification of an offset distance Δ. A region of graded composition (varying linearly along the direction normal to the interface) would be created using the compositions initially

assigned to either side of the interface ($\mathbf{m}^A$, $\mathbf{m}^B$) as the compositions to grade between:

$$\mathbf{m}^*_{final}(\mathbf{x}) = \begin{cases} \mathbf{m}^*_{fillet}(r) = \frac{1}{2}\left(\mathbf{m}^B + \mathbf{m}^A\right) + \frac{r}{2\Delta}\left(\mathbf{m}^B - \mathbf{m}^A\right) \text{ for } \mathbf{x} \in \text{ fillet region} \\ \mathbf{m}^*_{initial}(\mathbf{x}) \text{ otherwise} \end{cases} \tag{7.5}$$

The distance r is the signed distance of the point $\mathbf{x}$ from the interface into region 1. The second approach to creating a material fillet would begin with the specification of a maximum allowable rate of grading for each material ($Dm_i(\mathbf{x})\,\hat{\mathbf{n}}$) along with the selection of a face. In this case, the offset distance would be computed based on the difference between the compositions in the regions on either side of the interface and the limiting rates of material variation:

$$\Delta = \frac{1}{2}\max\left\{\frac{|m_0^A - m_0^B|}{\vec{\nabla}m_0(\mathbf{x})\cdot\hat{\mathbf{n}}}, \frac{|m_1^A - m_1^B|}{\vec{\nabla}m_1(\mathbf{x})\cdot\hat{\mathbf{n}}}, \ldots \frac{|m_{d_m-1}^A - m_{d_m-1}^B|}{\vec{\nabla}m_{d_m-1}(\mathbf{x})\cdot\hat{\mathbf{n}}}\right\} \tag{7.6}$$

Figure 7-15 illustrates the process of generating a material fillet.

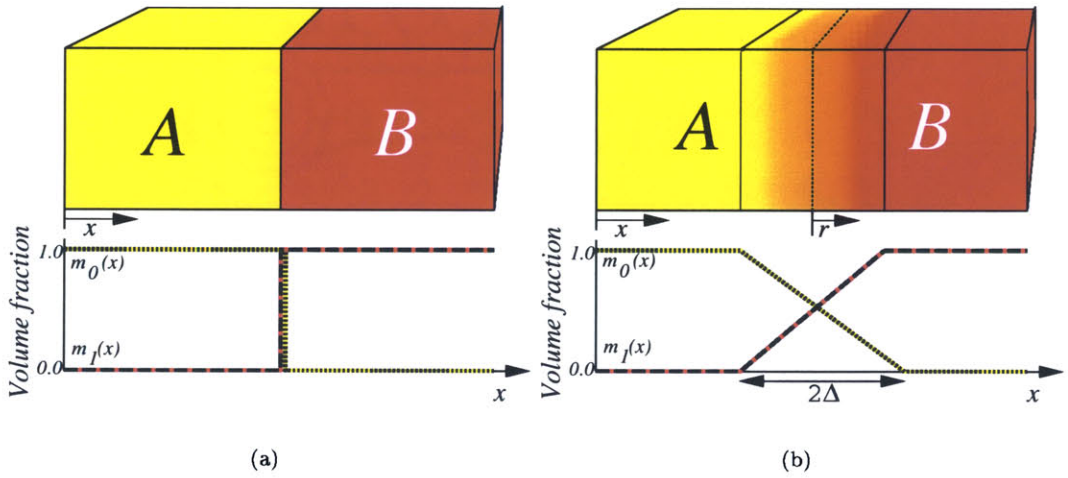


Figure 7-15: (a) Original material distribution over a cross-section of a block. (b) Material distribution over block cross-section with material fillet.

The final operation, material blend, is analogous to the previous two except that it generates an interface region with a composition that smoothly blends into the two neighboring regions ($M^1 continuous$). As with the material fillet, the thickness of the blend could either be defined explicitly or determined through the specification of a maximum rate of grading and material curvature.

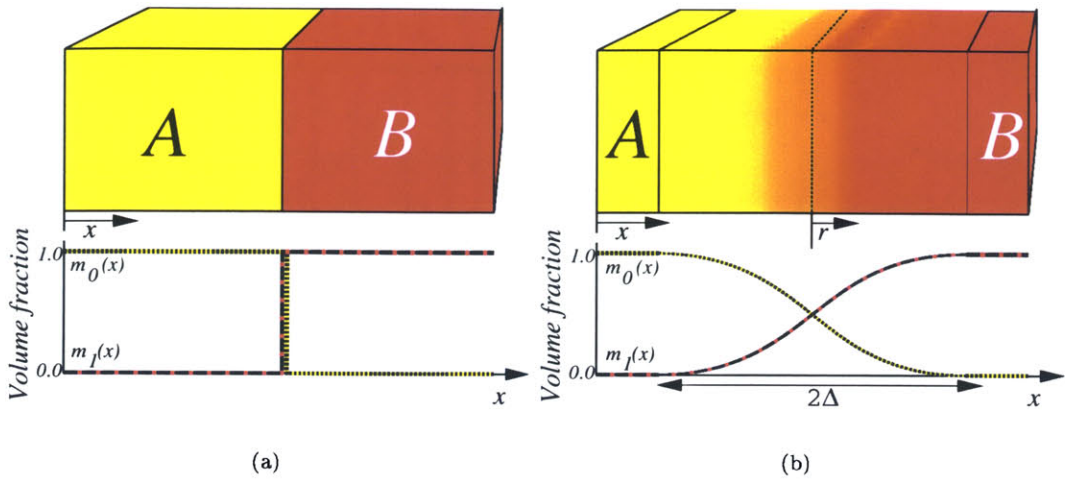


Figure 7-16: (a) Original material distribution over a cross-section of a block. (b) Material distribution over block cross-section with material blend.

7.5 Discussion

This chapter has proposed several methods to designing FGM objects. How these methods should be implemented is not addressed since this depends entirely on what underlying data structure is used to model the object. The introduction of these design methods, however, are proposed as pathways for capturing designer intent and as such, can be used for quantifying the storage cost for representing a FGM in terms of design intent.

It is also important to note that these design examples are direct adaptations of commonly used geometric tools in existing CAD systems. In this way, the design of material compositions can be seen as a logical extension of geometric design. Therefore, solutions for FGM design tools can be expected to be based on the same principles as used to define computer aided geometric design tools.

As a final note, the design tools presented here obviously do not represent an exhaustive but are only meant to motivate the investigation into FGM design methods. Additional design methods have also been proposed based on Boolean operations [40] and sweeps [53] for instance.

Chapter 8

The cost of representing composition

8.1 Motivation

Although several methods for defining the composition of an FGM object have been introduced, the basis for selecting one over the others as a preferred modeling method has not yet been established. One major factor in this decision is the memory required by each method to represent an object. To address this issue, this chapter introduces several hypothetical models illustrating issues that will be of relevance to modeling and designing real parts by the chosen modeling method. For the voxel-based, triangulated boundary, or tetrahedral mesh approaches, the idealized model must be discretized or approximated, as described in the preceding chapter. The resolution of this approximation is a function of the desired geometric and material accuracy, as well as the nature of the intended design. For each case, the FGM object is described and the expression for the storage cost is given for each approach. From this analysis, it is anticipated that justification for choosing one modeling approach over the others based on memory issues can be made. In order to produce the graphs showing the relative storage costs for the various data structures, the storage costs listed in Table 8.1 were assigned to the primitive data types. For the storage cost associated with representing a Material System, the symbol S_{ms} is used and its contribution to the overall cost is considered negligible ($S_{ms} = 0$ bytes) in graphs of the storage requirements for the various methods.

Data type	Symbol for storage cost per instance	Value used in analysis
integer	S_{int}	4 bytes
floating point number	S_{flt}	8 bytes
Boolean	S_{bln}	1 byte
pointer	S_{ptr}	4 bytes
Material System	S_{ms}	S_{ms}

Table 8.1: Storage costs associated with primitive data types on a Silicon Graphics O_2 workstation with 64 bit processor.

8.2 Case studies

8.2.1 Sphere of unit radius

The issue of geometric complexity, before even considering composition representation, requires a certain degree of overhead. For example, the boundaries of real mechanical parts often consist of smoothly blended surfaces with many features such as bosses, holes, fillets, and chamfers, requiring accurate descriptions of the surfaces and the features. One of the simplest cases to consider illustrating one of these factors is the sphere. Although trivial in its complexity, its representation by the various approaches to solid modeling highlight their fundamental differences in approach.

The first object to be considered is a sphere of unit radius ($R = 1\text{mm}$) positioned at the origin, consisting of a single material. The only information that needs to be conveyed is the nature of its boundary between the object interior and exterior, as illustrated in Figure 8-1. For uniformity in analysis with subsequent cases, a two dimensional Material Space ($d_m = 2$) is used, in which one material is the interior of the sphere and the second material is the empty (void) space surrounding the object.

$$\mathbf{m}^*(\mathbf{x}) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{for } ||\mathbf{x}|| \leq 1; \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{otherwise.} \end{cases} \tag{8.1}$$

Three methods of representing the sphere are analyzed here: voxel-based approach, triangulated boundary, and the generalized decomposition methods (Cell-Tuple-Graph and Radial-Edge). The sphere's representations in the various data structures are illustrated in Figures 8-2, 8-3, and 8-4.

Voxel-based representation of a sphere

The first method considered here is the voxel-based data structure, as illustrated in Figure 8-2, in which the object is discretized into a lattice of voxels. In this modeling method, the storage cost is a factor of the physical size of the object as well as the desired resolution of the geometry and

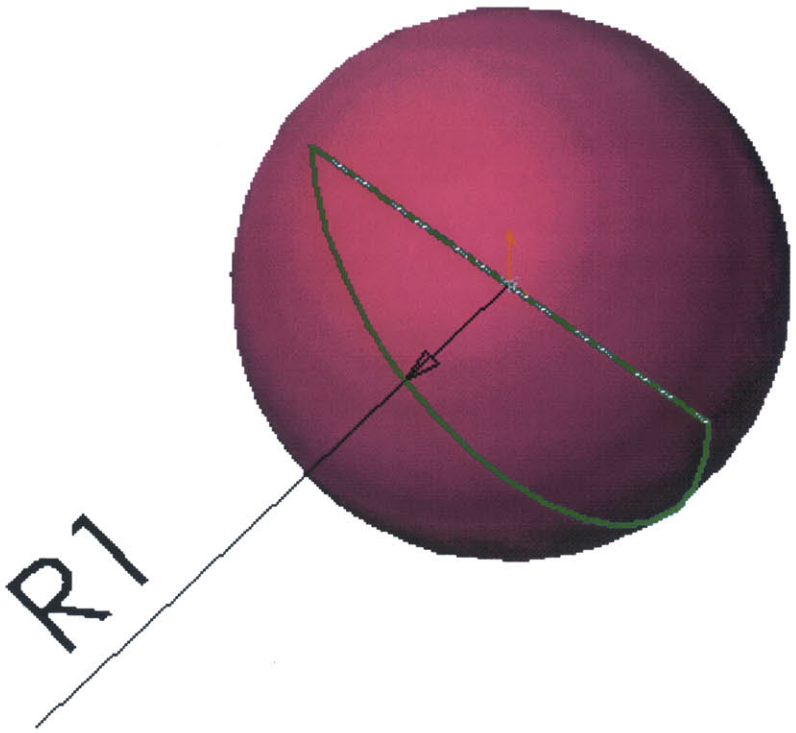


Figure 8-1: Geometric design of unit sphere.

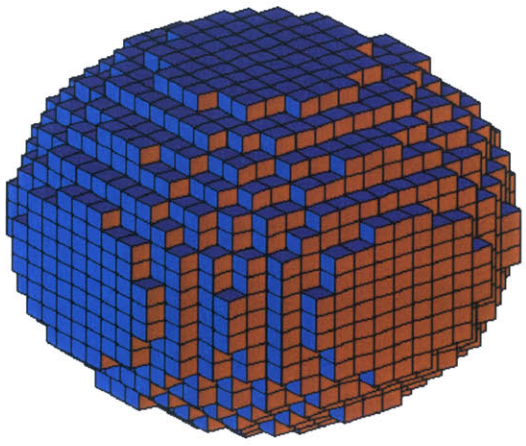


Figure 8-2: Voxelized approximation of sphere.

composition. Since this model consists of a uniform composition, only a binary value needs to be associated with each voxel to indicate the presence or not of material ($n_\lambda = 2$). Substituting these values into the expression for the storage cost of the voxelized method (Equation 5.22), the cost associated with modeling the unit sphere as a voxelized model is:

$$\begin{aligned} S_{vox}[L_x = 2mm, L_y = 2mm, L_z = 2mm, & \\ \epsilon_g, n_\lambda = 2, d_m = 2] &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{1}{4}\left\lceil \frac{2}{\epsilon_g} \right\rceil^3 \\ &= 64 + \frac{1}{4}\left\lceil \frac{2}{\epsilon_g} \right\rceil^3 + S_{ms} \text{ bytes} \end{aligned}$$

The parameter ϵ_g is the geometric accuracy in modeling the intended spherical boundary of the object.

Triangulated boundary

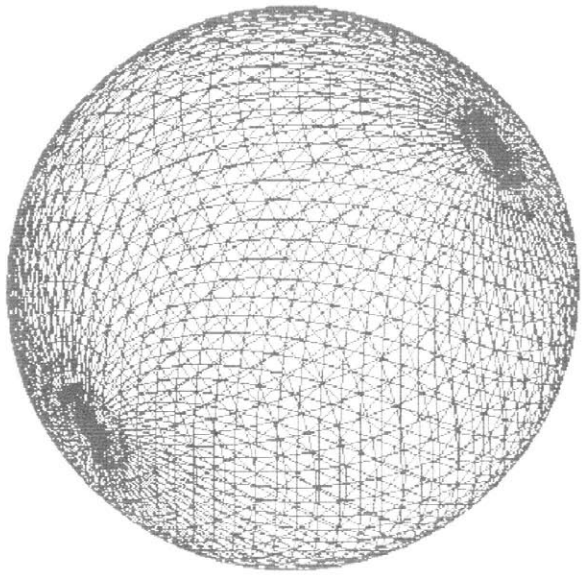


Figure 8-3: Triangulation of sphere.

The next level of sophistication in modeling the boundary of the sphere is the use of a triangulated mesh, as shown in Figure 8-3. Through the approach presented in Section 4.4, the sphere would be represented by two regions, corresponding to the spaces interior and exterior to the sphere's boundary. Each region references the mesh of triangles approximating the desired, curved boundary, as well as a uniform composition assigned to the region.

The cost for representing a model in terms of a triangulated mesh is directly dependent on the number of triangles in the model. According to Equation 6.34, the number of triangles required to achieve a desired geometric accuracy is a function of the surface area, surface curvature, and the

minimum feature size. For the model of the sphere, the number of triangles in the boundary is:

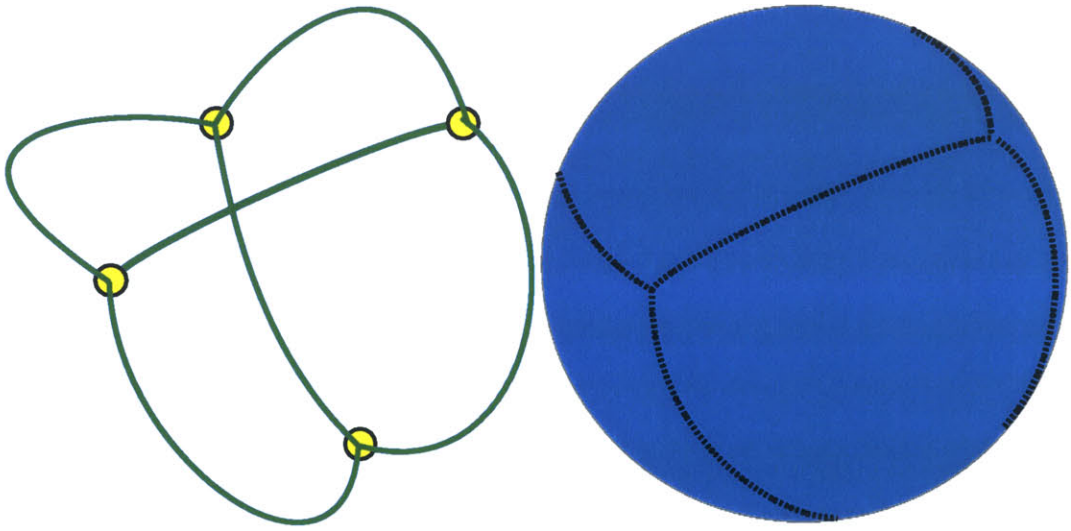
$$n_{triangles}[A_{surface} = 4\pi mm^2, \epsilon_g, \kappa_{g,max} = 1mm^{-1}, \mu_g = 2mm] = O\left\lceil \frac{16\sqrt{3}\pi}{3a^2} \right\rceil$$
$$\text{where } a = \min\left\{2, \sqrt{3}\left(\arcsin\sqrt{\epsilon_g(1-\epsilon_g)}\right)\right\} \tag{8.2}$$

Substituting the above into Equation 6.33, the storage cost for the unit sphere as a triangulated mesh in terms of geometric accuracy is formed:

$$\begin{aligned} S_{tri}[n_{regions} = 2, a] &\leq 3S_{int} + 2S_{ptr} + 4S_{flt} + S_{ms} + (5S_{ptr} + 2S_{bln} + \frac{3}{2}S_{flt})\left\lceil \frac{16\sqrt{3}\pi}{3a^2} \right\rceil \\ &\leq 52 + 34\left\lceil \frac{16\sqrt{3}\pi}{3a^2} \right\rceil + S_{ms} \text{ bytes} \\ &\quad \text{where } a = \min\left\{2, \sqrt{3}\left(\arcsin\sqrt{\epsilon_g(1-\epsilon_g)}\right)\right\} \end{aligned} \tag{8.3}$$

As $\epsilon_g \to 0$, the geometric accuracy is the dominant factor in determining the number of triangular facets in the mesh and $a \sim (3\epsilon_g)^{\frac{1}{2}}$ and $n_{triangles} \sim \frac{(2)^4\pi}{(3)^{\frac{3}{2}}(\epsilon_g)}$. For geometrically accurate models (small ϵ_g), the storage cost for the triangulated mesh modeling method grows as $S_{tri} \sim \frac{(2)^5 17\pi}{(3)^{\frac{3}{2}}(\epsilon_g)} = O\left(\epsilon_g^{-1}\right)$.

Generalized decomposition



(a) (b)

Figure 8-4: (a) Vertices and edges in generalized representation of sphere. (b) Faces in generalized representation of sphere.

To represent the curved nature of the sphere's boundary exactly, it is necessary to use a nonlinear,

rational definition for the modeled surfaces. The Radial Edge and Cell-Tuple-Graph data structures permit this, representing the shape of the model in terms of a set of FGMDomains that are related to each other in a generalized, topological database. Using the FGMDomains defined in Section 4.6.2, the sphere can be decomposed into two generalized regions (its interior and its complement, the exterior), four rational Bézier triangles, six rational Bézier curves, and four vertices, as shown in Figures 8-4(a) and (b). The FGMDomains used to model the sphere and their contributed costs are listed in Table 8.2, along with the cost associated with representing the topology within the Cell-Tuple-Graph data structure.

Classes	*instances*	$\frac{ctgs}{class}$	$\frac{cells}{class}$	$\frac{tuples}{class}$	$\frac{Storage(bytes)}{class}$
FGMPoint	4		4		$20S_{flt}$
FGMRationalBézierCrv $n_x = 2, n_m = 0$	6		6		$12S_{int} + 90S_{flt}$
FGMRationalBézierTri $n_x = 2, n_m = 0$	4		4	48	$8S_{int} + 108S_{flt}$
FGMBRepRegion	2	1	2		$4S_{flt} + 2S_{ptr}$
CellTupleGraph		1			$2S_{ptr}$
Cell			16		$32S_{int} + 32S_{ptr} + S_{ms}$
Tuple				48	$48S_{int} + 432S_{ptr}$
TOTAL					$100S_{int} + 222S_{flt} + 468S_{ptr} + S_{ms}$ $= 4048$ bytes $+S_{ms}$

Table 8.2: FGMDomain and Cell-Tuple-Graph objects required to represent sphere object exactly and the associated storage cost.

For the same set of FGMDomains, the cost associated with representing the object within the Radial-Edge data structure is given in Table 8.3.

Classes	*instances*	$\frac{Storage(bytes)}{class}$
FGMDomains		$20S_{int} + 222S_{flt} + 2S_{ptr}$
Complex	1	$S_{ptr} + S_{ms}$
Region	2	$10S_{ptr}$
Shell	2	$8S_{ptr}$
Face	4	$8S_{ptr}$
Loop	4	$4S_{ptr}$
Edge	6	$12S_{ptr}$
Vertex	4	$8S_{ptr}$
FaceUse	8	$48S_{ptr}$
LoopUse	8	$48S_{ptr}$
EdgeUse	24	$168S_{ptr}$
VertexUse	24	$96S_{ptr}$
TOTAL		$20S_{int} + 222S_{flt} + 413S_{ptr} + S_{ms}$
		3508 bytes $+S_{ms}$

Table 8.3: Number of instances of Radial-Edge objects required to represent sphere model exactly and the associated storage cost.

Graphically, the storage cost for the sphere modeled within these data structures is illustrated

in Figure 8-5.

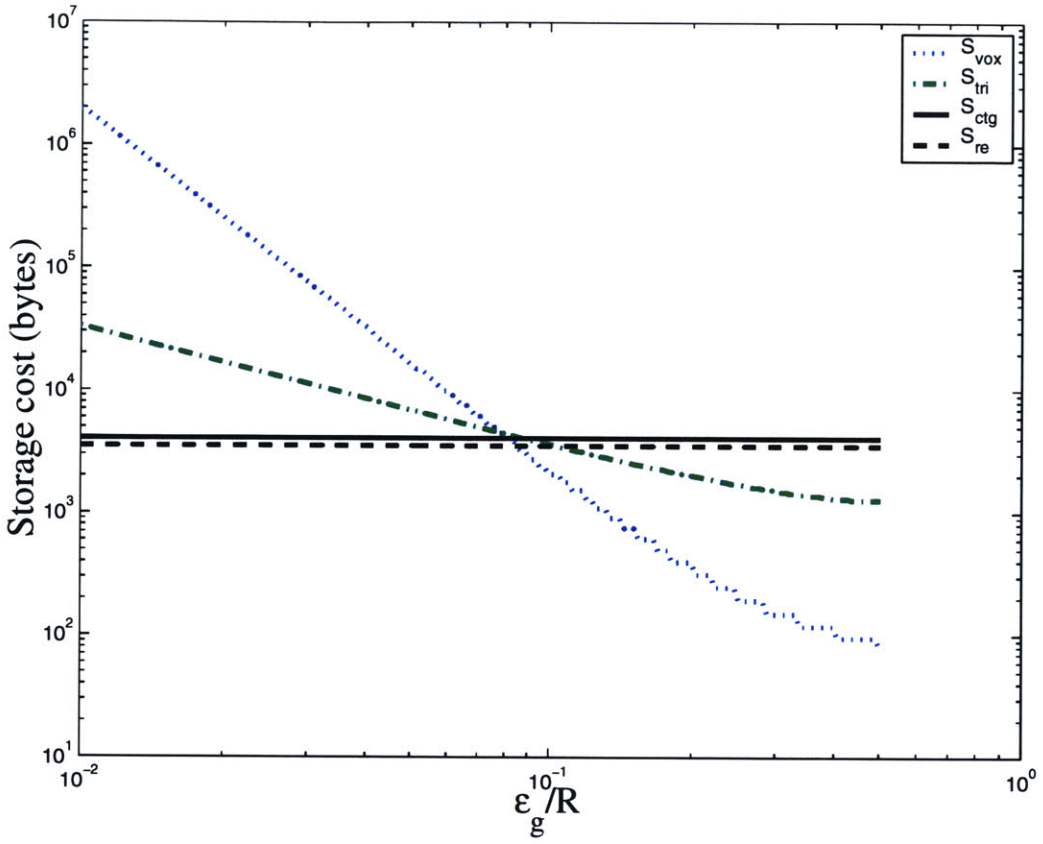


Figure 8-5: Graph of storage cost for representing a unit sphere as a function of geometric accuracy for the data structures considered.

Although the sphere is a trivial example, it serves to demonstrate the differences between the representation methods considered in this dissertation. In the current implementation, the storage costs associated with representing the sphere in the STL file specification and the STEP standard are illustrated in Figure 8-6.

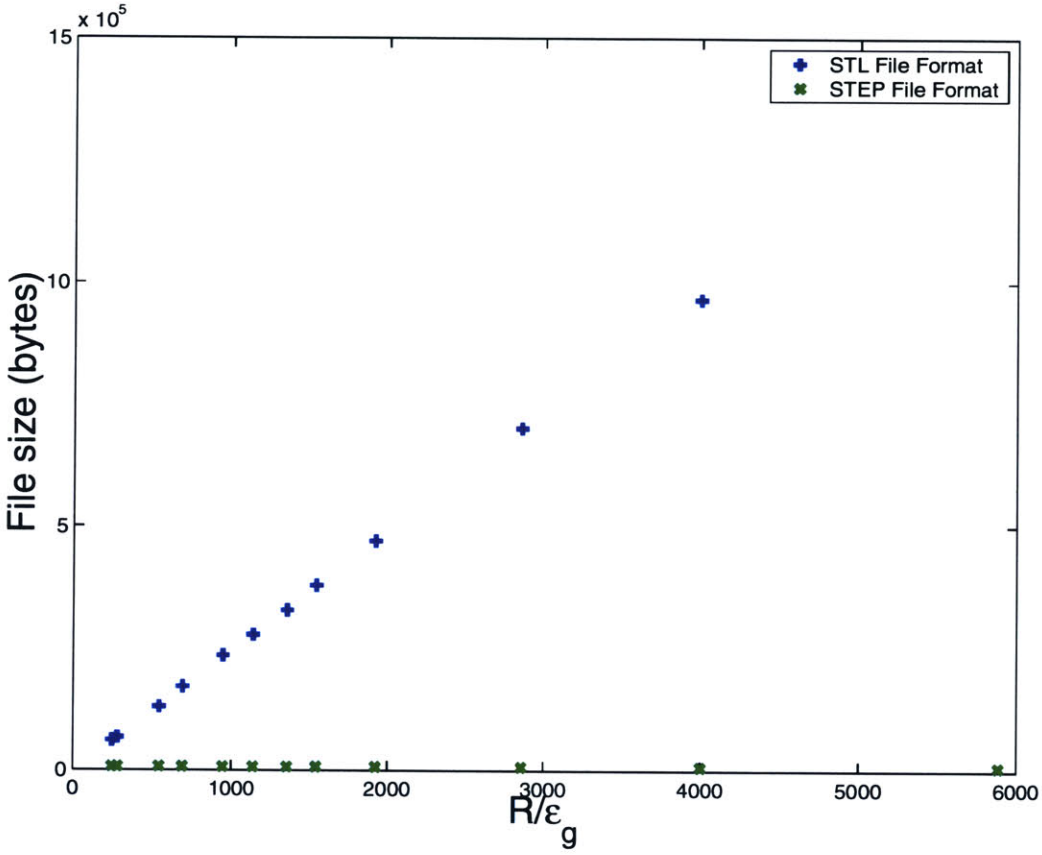


Figure 8-6: Graph of storage cost for representing a sphere as a function of geometric accuracy in the STL and STEP file formats.

8.2.2 Bar with graded transition

With the capability of Local Composition Control, the intended composition of an object may not consist solely of uniform or linearly varying composition, but may contain complex, graded regions defined according to higher order functions. This is analogous to the geometric boundaries of real mechanical parts containing curved and freeform surface patches and features. As the previous example represented a simple, curved model for geometric considerations, one of the simplest designs demonstrating a graded composition is an FGM bar consisting of two regions of uniform composition at either end with a smoothly graded interface region between the two. With this as a case study, the storage costs for the various modeling methods as functions of material accuracy are studied.

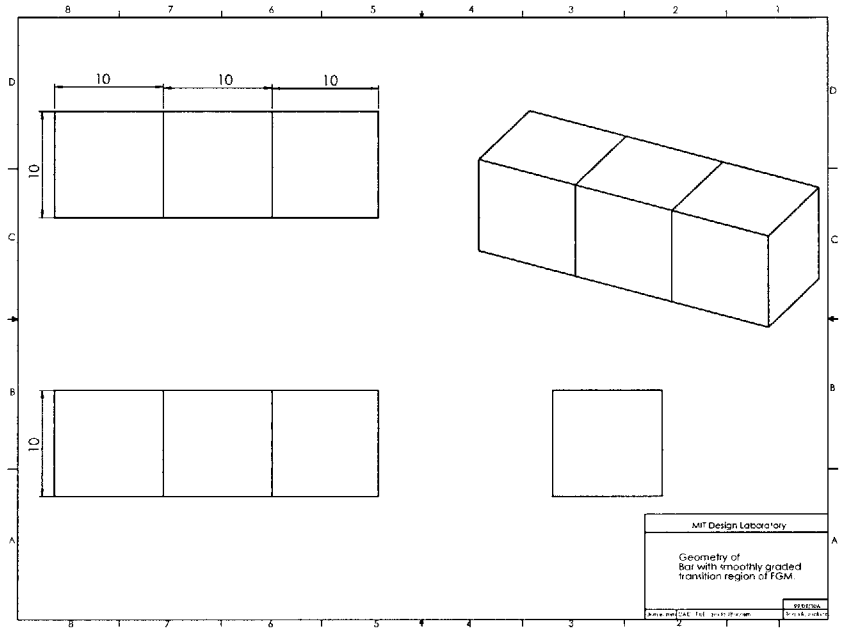


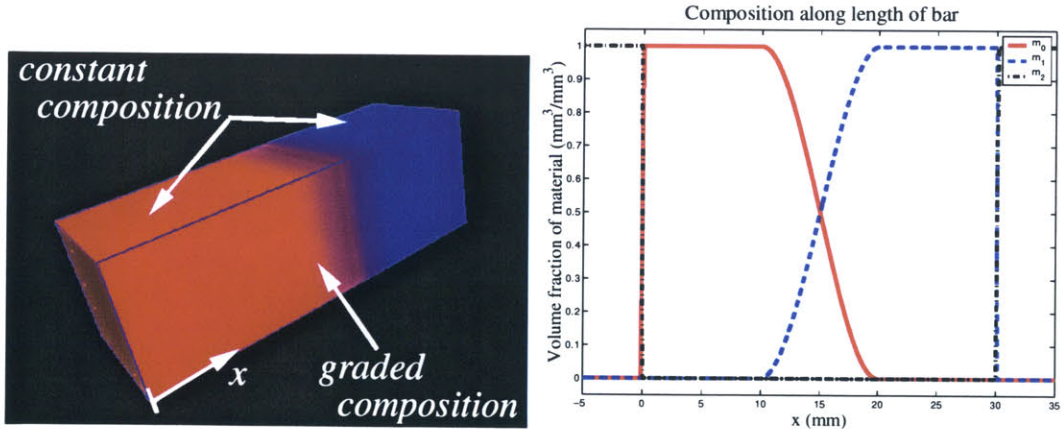
Figure 8-7: Geometric bar specimen to contain smoothly graded transition between two different compositions.

Figure 8-7 shows the geometric design of the bar. It is divided into three equal segments along its length, each 10mm long. The intended design consists of two materials plus voids, placing the model in a three dimensional Material Space ($d_m = 3$). The composition at either end is uniform while the

interface region smoothly grades according to a cubic function. This composition is expressed as:

$$\mathbf{m}^*(\mathbf{x}) = \begin{cases} \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} & \text{for } 0 \leq x \leq 10,\ 0 \leq y \leq 10,\ 0 \leq z \leq 10 \\ \begin{bmatrix} 1.0 - \frac{3}{100}(x-10)^2 + \frac{1}{500}(x-10)^3 \\ \frac{3}{100}(x-10)^2 - \frac{1}{500}(x-10)^3 \\ 0.0 \end{bmatrix} & \text{for } 10 < x < 20,\ 0 \leq y \leq 10,\ 0 \leq z \leq 10 \\ \begin{bmatrix} 0.0 \\ 1.0 \\ 0.0 \end{bmatrix} & \text{for } 20 \leq x \leq 30,\ 0 \leq y \leq 10,\ 0 \leq z \leq 10 \\ \begin{bmatrix} 0.0 \\ 0.0 \\ 1.0 \end{bmatrix} & \text{for } x \text{ otherwise} \end{cases} \tag{8.4}$$

The coordinate system for the bar and the graded composition given above are illustrated graphically in Figure 8-8.



(a) (b)

Figure 8-8: (a) Desired decomposition of bar into uniform and graded regions. (b) Graded composition along length of bar.

Three different classes of methods for capturing the designer's intent are compared here: voxel-based, tetrahedral mesh, and generalized decomposition (Cell-Tuple-Graph and Radial-Edge). The

information represented by each of these methods is shown in Figures 8-9, 8-10, and 8-11. To simplify the following analysis, the geometric accuracy is treated as constant, with $\epsilon_g = 1\text{mm}$. In this way, the variation in the storage requirements will depend solely on the desired material accuracy ϵ_m for representing the intended material variation.

Voxel-based representation of a graded bar

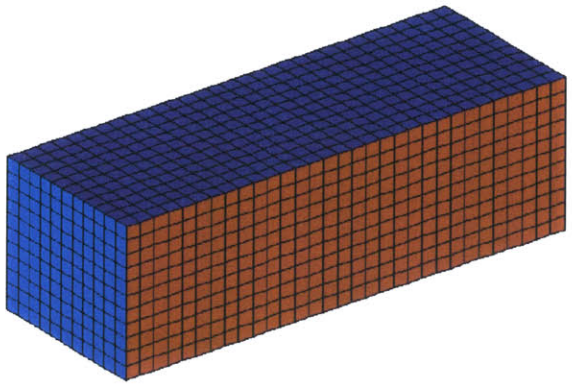


Figure 8-9: Voxelized approximation of bar.

Figure 8-9 illustrates the FGM bar decomposed into voxels. According to Section 5.6, the factors affecting the storage cost of the voxelized representation included the physical size of the model, the desired resolution in composition (geometric accuracy is considered constant), minimum feature sizes (μ_g and μ_m) and the nature of the desired grading ($\mathbf{m}^*(\mathbf{x})$):

$$\begin{aligned} S_{vox} &= S_{vox}[L_x = 10\text{mm}, L_y = 10\text{mm}, L_z = 10\text{mm}, \\ &\qquad \epsilon_g = 1\text{mm}, \epsilon_m, M^*, \mu_g = 10\text{mm}, \mu_m = 10\text{mm}] \end{aligned} \tag{8.5}$$

The minimum geometric features size is simply the width or height of the bar ($\mu_g = 10\text{mm}$). The minimum material feature size, defined as the minimum distance between two discontinuities in composition or its derivative, is the same as the minimum geometric feature size since the variation of the composition within the bar is smooth. The parameter M^* is a function of the desired grading, as defined in Equation 5.21. For the model considered here, M^* would be defined by either material m_0 or m_1 over the plane $\pi : x = 15$. Along the direction parallel to the length of the bar, $\hat{\mathbf{v}} = [1\ 0\ 0]$,

the rate of material variation is the greatest.

$$\begin{aligned} M^* &= \left|\vec{\nabla} m_0^*(\mathbf{x}) \cdot \hat{\mathbf{v}}\right| \\ &= \left|\left[\tfrac{-3}{50}(x-10) + \tfrac{3}{500}(x-10)^2 \quad 0 \quad 0\right] \cdot \hat{\mathbf{v}}\right| \\ &= \left|\left[\tfrac{-3}{50}(15-10) + \tfrac{3}{500}(15-10)^2 \quad 0 \quad 0\right] \cdot \begin{bmatrix}1 & 0 & 0\end{bmatrix}\right| \\ &= \frac{3}{20} mm^{-1} \end{aligned}$$

Substituting the values for minimum feature size and M^* into Equation 5.21, an expression for the memory required to model the FGM bar as a function of material accuracy is formed:

$$\begin{aligned} S_{vox} &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{3}{8}\left\lceil\frac{30}{a}\right\rceil\left\lceil\frac{10}{a}\right\rceil^2\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil \\ &= 64 + \frac{3}{8}\left\lceil\frac{30}{a}\right\rceil\left\lceil\frac{10}{a}\right\rceil^2\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil + S_{ms} \qquad (8.6) \\ &\quad \text{where } a = \min\{10^{-1}\text{mm}, \frac{40\sqrt{3}\epsilon_m}{9}\} \end{aligned}$$

Tetrahedral mesh

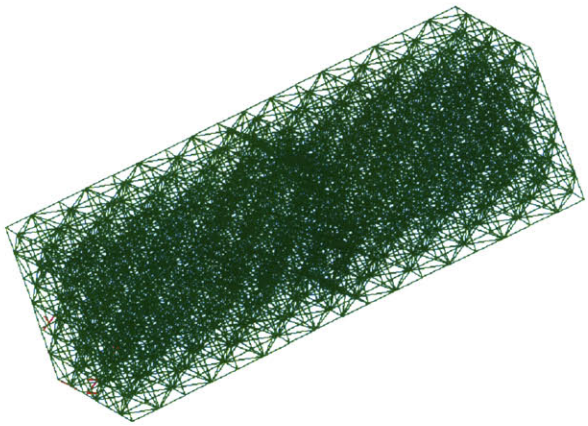


Figure 8-10: Tetrahedral mesh approximation of bar.

The next modeling method to be considered for modeling the FGM bar is the tetrahedral mesh scheme explained in Section 4.5. As was discussed in Section 6.7, the storage requirements for the various tetrahedral databases presented are all dependent on the number of tetrahedra in the mesh. The number of tetrahedra, in turn, is a function of the desired geometry ($V_{interior}$, μ_g, and κ_g), the desired composition (μ_{mt} and κ_m), as well as the geometric and material accuracy. Again, the geometric accuracy is considered constant ($\epsilon_g = 10^{-1}$mm) to simplify the analysis. As was the case for the voxelized analysis, the minimum feature sizes are equal to the width of the model: $\mu_g = \mu_{mt} = 10$mm. Since the model consists only of planar facets, the geometric curvature is zero

($\kappa_g = 0$). The maximum material curvature, however, is nonzero since the desired composition is graded according to a cubic polynomial for materials m_0 and m_1 through the interface region. For material m_0, the material curvature, along the direction of grading ($\mathbf{v} = [1\ 0\ 0]$), over the interface region ($10 \leq x \leq 20$) is given by Equation 6.24:

$$\begin{aligned}
\kappa_{m_0}\Big|_{\hat{\mathbf{v}}=[1\ 0\ 0]} &= \frac{\vec{\nabla}[\vec{\nabla} m_0^*(\mathbf{x}) \cdot \hat{\mathbf{v}}] \cdot \hat{\mathbf{v}}}{\left[1 + \left(\vec{\nabla} m_0^*(\mathbf{x}) \cdot \hat{\mathbf{v}}\right)^2\right]^{\frac{3}{2}}} \\
&= \frac{\vec{\nabla}\left\{\vec{\nabla}\left[1.0 - \frac{3}{100}(x-10)^2 + \frac{1}{500}(x-10)^3\right] \cdot [1\ 0\ 0]\right\} \cdot [1\ 0\ 0]}{\left[1 + \left(\vec{\nabla}\left(1.0 - \frac{3}{100}(x-10)^2 + \frac{1}{500}(x-10)^3\right) \cdot [1\ 0\ 0]\right)^2\right]^{\frac{3}{2}}} \\
&= \frac{\frac{\partial^2}{\partial x^2}\left[1.0 - \frac{3}{100}(x-10)^2 + \frac{1}{500}(x-10)^3\right]}{\left[1 + \left(\frac{\partial}{\partial x}\left(1.0 - \frac{3}{100}(x-10)^2 + \frac{1}{500}(x-10)^3\right)\right)^2\right]^{\frac{3}{2}}} \\
&= \frac{3x - 45}{250\left[\left\{1 + \left[\frac{9(x-10)^4 + 180(x-10)^3 + 9000(x-10)^2}{250000}\right]^2\right\}^{\frac{3}{2}}\right.} \\
&\qquad \text{for } 10 \leq x \leq 20,\ 0 \leq y \leq 10,\ 0 \leq z \leq 10
\end{aligned} \tag{8.7}$$

The material curvature is greatest over the planes $\pi_l : x = 10mm$ and $\pi_r : x = 20mm$, where $\kappa_{m,max} = \frac{3}{50}\text{mm}^{-2}$. From this information, a bound for the number of tetrahedra to achieve the desired geometric and material accuracy can be formed using Equation 6.41

$$\begin{aligned}
n_{tetrahedra}[V_{interior} = 3000mm^3, \epsilon_g = 10^{-1}mm,& \\
\mu_g = 10mm, \mu_{mt} = 10, \kappa_{m,max} = \frac{3}{50}mm^{-2}] &= O\left[\frac{18000\sqrt{2}}{a^3}\right] \\
\text{where } a = \min&\left\{\frac{100\arcsin\left(\sqrt{\frac{3}{50}\epsilon_m(2 - \frac{3}{50}\epsilon_m)}\right)}{3},\ 10\right\}.
\end{aligned}$$

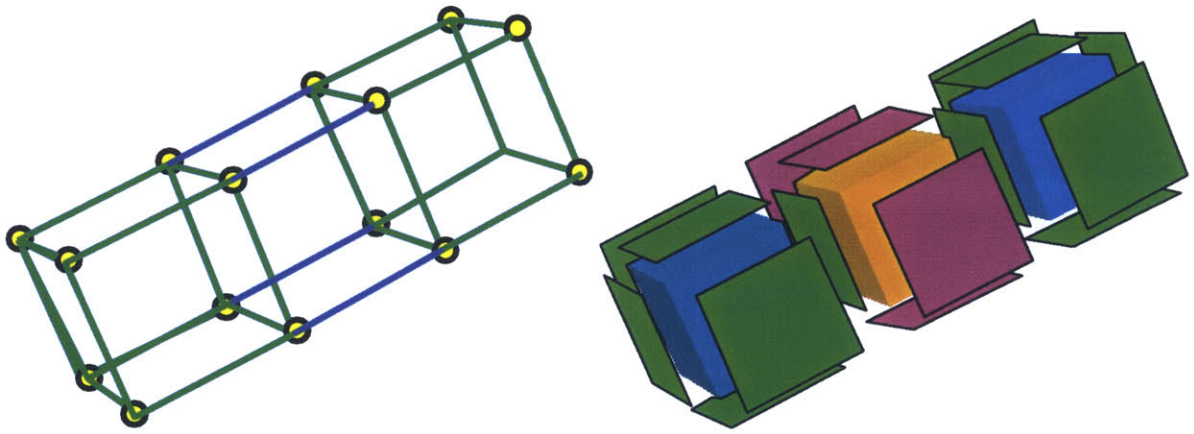
As $\epsilon_m \to 0$, the desired material accuracy dominates in the above expression and $a \sim \frac{4}{3}\sqrt{3\epsilon_m}$ and $n_{tetrahedra} \sim 5^3\left(\frac{3}{2\epsilon_m}\right)^{\frac{3}{2}} = O\left(\epsilon_m^{-\frac{3}{2}}\right)$.

With a bound for the number of tetrahedra known, a bound for the storage requirements for the

various tetrahedral modeling schemes can be established for this object (Equations 6.35- 6.40):

$$
\begin{aligned}
S_{te_{00}} &\leq (4S_{ptr} + 6S_{flt})n_{tetrahedra} + 18S_{flt} + S_{int} + S_{ptr} + S_{ms} \\
&\leq 64n_{tetrahedra} + 152 + S_{ms} \text{ bytes} \\
S_{te_{01}} &\leq (4S_{ptr} + 15S_{flt})n_{tetrahedra} + 9S_{flt} + S_{int} + S_{ptr} + S_{ms} \\
&\leq 136n_{tetrahedra} + 80 + S_{ms} \text{ bytes} \\
S_{te_{10}} &\leq (8S_{ptr} + 6S_{flt} + S_{int})n_{tetrahedra} + 18S_{flt} + 4S_{int} + S_{ptr} + S_{ms} \\
&\leq 84n_{tetrahedra} + 164 + S_{ms} \text{ bytes} \\
S_{te_{11}} &\leq (8S_{ptr} + 15S_{flt} + S_{int})n_{tetrahedra} + 9S_{flt} + 4S_{int} + S_{ptr} + S_{ms} \\
&\leq 156n_{tetrahedra} + 92 + S_{ms} \text{ bytes}
\end{aligned}
$$

Generalized decomposition



(a) (b)

Figure 8-11: (a) Vertices and edges in generalized representation of bar. (b) Faces and regions in generalized representation of bar.

The final two methods for representing the FGM bar are the Cell-Tuple-Graph and the Radial-Edge data structures. For these representation methods, the object is subdivided in FGMDomains as shown in Figures 8-11(a) and (b). The model of the bar consists of two piece-wise constant, B-rep FGMDomains, a single Hexahedral FGMDomain, and the corresponding vertices, edges, and faces. The intended material design is represented exactly by the use of an FGMDomain region with cubic blending in the interface region. The number of instances and the associated storage cost for the FGMDomains are tabulated in Table 8.4, along with the number of instances of Cell-Tuple-Graph objects and the total cost for the representation of the bar within the Cell-Tuple-Graph data structure. The cost associated with the Radial-Edge data structure is listed in Table 8.5.

Classes	*instances*	$\frac{ctgs}{class}$	$\frac{cells}{class}$	$\frac{tuples}{class}$	$\frac{Storage(bytes)}{class}$
FGMPoint	16		16		$96S_{flt}$
FGMRationalBézierCrv					
$n_x = 1, n_m = 0$	24		24		$48S_{int} + 288S_{flt}$
$n_x = 1, n_m = 3$	4		4		$8S_{int} + 96S_{flt}$
FGMRationalBézierQuad					
$(m_x, n_x) = (1,1),\ (m_m, n_m) = (0,0)$	12		12	192	$48S_{int} + 240S_{flt}$
$(m_x, n_x) = (1,1),\ (m_m, n_m) = (0,3)$	4		4	64	$16S_{int} + 128S_{flt}$
FGMRationalBézierHex					
$(l_x, m_x, n_x) = (1,1,1),$					
$(l_m, m_m, n_m) = (0,0,3)$	1		1		$6S_{int} + 48S_{flt}$
FGMBRepRegion					
with 1 boundary	3	1	3		$9S_{flt} + 3S_{ptr}$
CellTupleGraph		1			$2S_{ptr}$
Cell			64		$128S_{int} + 128S_{ptr} + S_{ms}$
Tuple				256	$256S_{int} + 2304S_{ptr}$
TOTAL					$510S_{int} + 905S_{flt} + 2437S_{ptr} + S_{ms}$
					19028 bytes + S_{ms}

Table 8.4: FGMDomain and Cell-Tuple-Graph objects required to represent bar object exactly and the associated storage cost.

Classes	*instances*	$\frac{Storage(bytes)}{class}$
FGMDomains	$126S_{int} + 905S_{flt} + 3S_{ptr}$	
Complex	1	$S_{ptr} + S_{ms}$
Region	4	$20S_{ptr}$
Shell	4	$16S_{ptr}$
Face	16	$32S_{ptr}$
Loop	16	$16S_{ptr}$
Edge	28	$56S_{ptr}$
Vertex	16	$32S_{ptr}$
FaceUse	32	$192S_{ptr}$
LoopUse	32	$192S_{ptr}$
EdgeUse	128	$896S_{ptr}$
VertexUse	128	$512S_{ptr}$
TOTAL	$126S_{int} + 905S_{flt} + 1968S_{ptr} + S_{ms}$	
	15616 bytes + S_{ms}	

Table 8.5: The Radial-Edge objects required to represent FGM bar object exactly and the associated storage cost.

Graphically, the storage costs for the various methods for representing a FGM bar with one dimensional, cubic grading is illustrated in Figure 8-12.

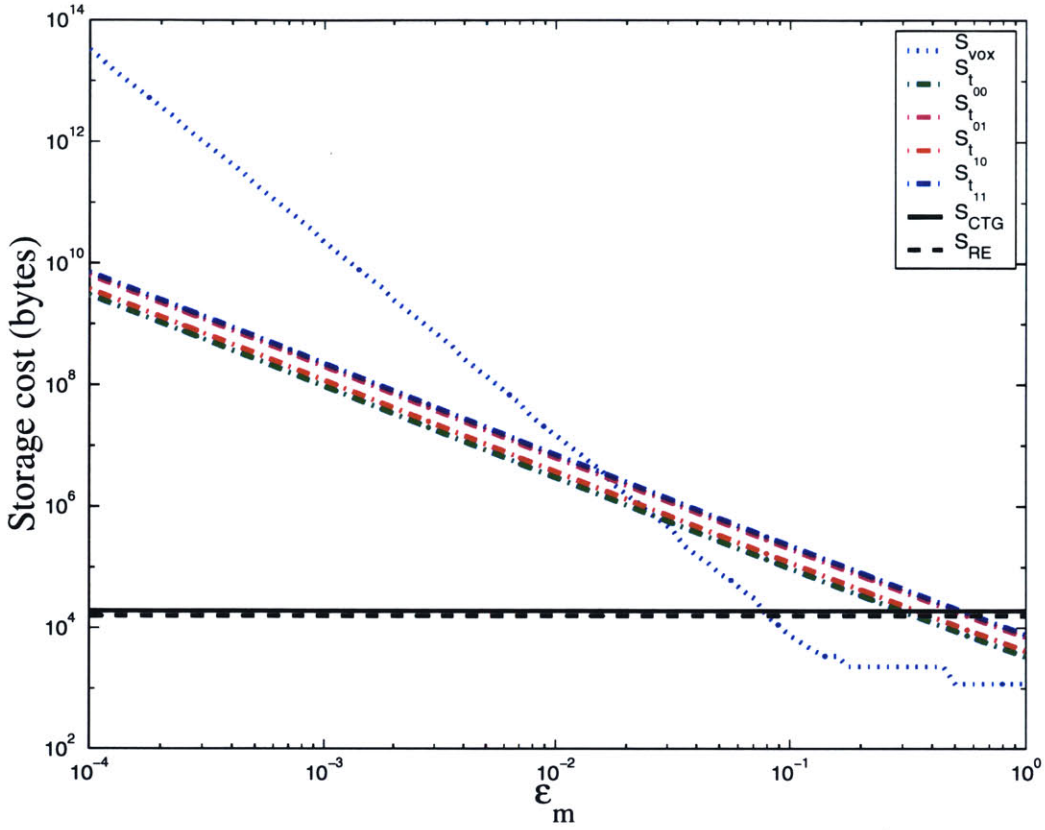


Figure 8-12: Graph of storage cost for representing an FGM bar specimen as a function of material accuracy within the indicated data structures with $\epsilon_g = 0.1$mm.

8.2.3 Graded composition from boundary of cavity in block

Whereas the first two cases addressed nonlinearities in shape and composition separately, more general models may contain both nonlinear geometric elements and nonlinear material gradings. As a first example, the design of a block with a cavity is presented, as shown in Figure 8-13, containing both filleted features and nonlinearly varying compositions.

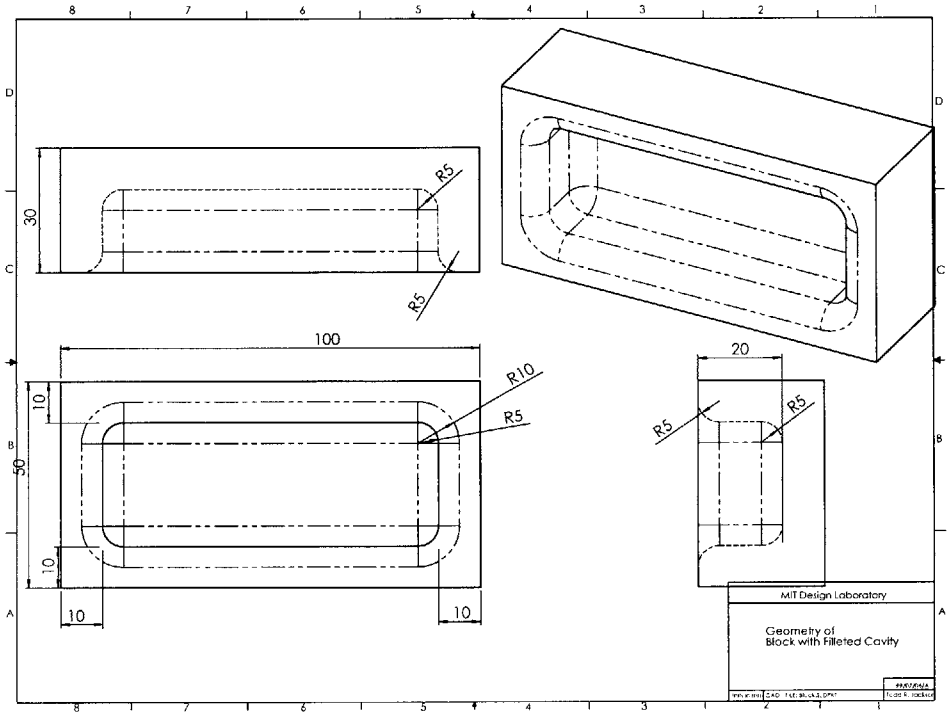
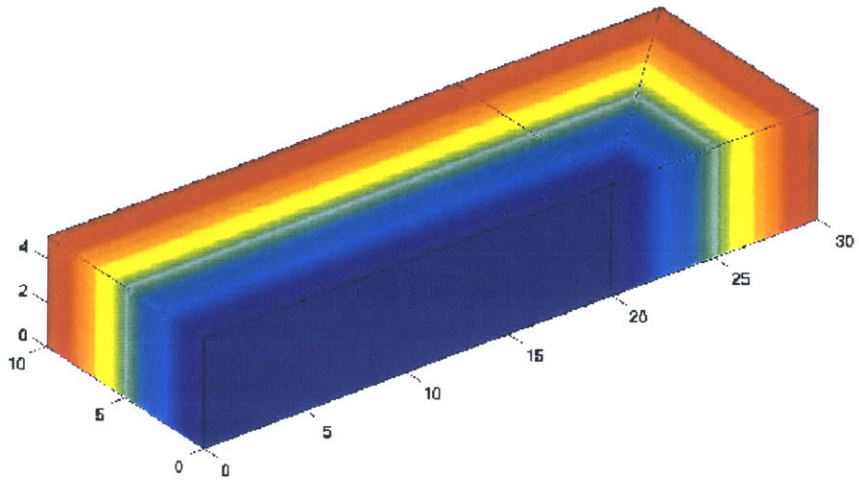


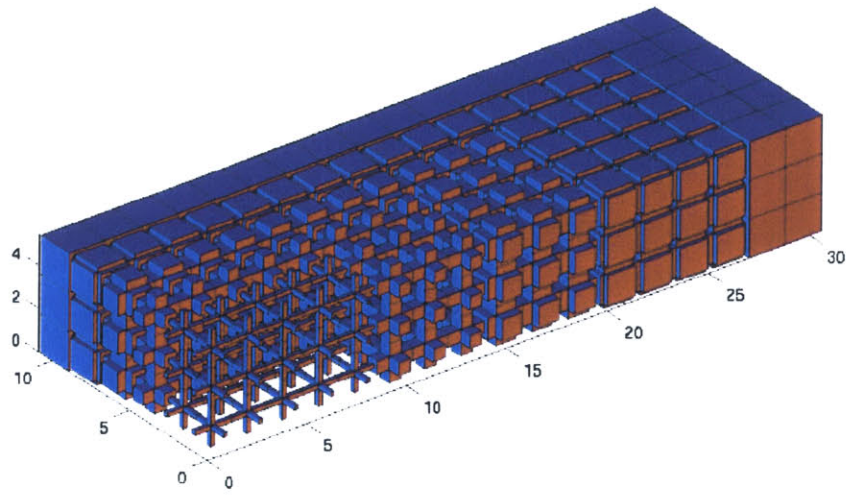
Figure 8-13: Geometric design of block with cavity.

The object in Figure 8-13 represents a generic mold: a block with a cavity into which molten material can be poured and solidified to form a part. It has been hypothesized that the thermal inertia of a mold can be reduced by designing molds with internal cavities and passages [59], thereby reducing the cooling time and increasing the cycle time for manufacturing parts. With the concept of locally controlling compositions with porosity, the representation of such parts may be efficiently handled by a suitable FGM modeling representation. To illustrate this concept, consider the model in Figure 8-13 to be material designed within a two dimensional Material Space. The first material is the solid material (m_0) out of which the mold is fabricated and the second material is void space (m_1), allowing the representation of porosity. By designing a composition that grades from fully dense material m_0 at the walls of the mold to some porosity over some distance from the corresponding surfaces, the role of the mold cavity to define the shape of the molded part is preserved while decreasing the total mass of the mold, thereby reducing the mold's thermal inertia.

The porosity in the fabricated part is achieved through mapping the composition from the con-



(a)



(b)

Figure 8-14: (a) Intended density distribution over bar specimen, grading of fully dense material at the surfaces $x = 30mm$ and $y = 10mm$ to 20% density at a distance of 10mm from these boundaries. (b) View of halftoned bar illustrating porous macro-structure generated through the halftoning of the continuous FGM model into binary material primitives.

tinuous material volume fraction values ($\mathbf{m}(\mathbf{x})$) represented in the FGM model to discrete macro-structures using a suitable dithering lattice. If the dithering lattice satisfies the requirement that adjacent resulting macro-structures are contiguous, the resulting structure will be structural solid but containing the porosity. This concept is illustrated for the bar in Figure 8-14(a) with a composition graded linearly from full density material at the surfaces $x = 30$mm and $y = 10$mm. In order to convert the continuous value into a discrete value, a $9 \times 9 \times 9$ dithering lattice was used, as defined by:

$$\begin{aligned} \mathbf{D} &= \{D_{ijk} \text{ for } 0 \leq i < 9,\ 0 \leq j < 9,\ 0 \leq k < 9\}, \\ &\text{where } D_{ijk} = \frac{\min\{|i-4|, |j-4|, |k-4|\} + 1}{5} \end{aligned} \tag{8.8}$$

The intended composition variation ($\mathbf{m}^*(\mathbf{x})$) for the bar was sampled over a lattice of points ($\mathbf{x_i}$). The intensity of the material was then compared with the threshold values in the dither lattice. Wherever the intensity was greater than the corresponding value in the dither lattice, a material primitive was placed[1]. The resulting structure after thresholding is illustrated in Figure 8-14(b), demonstrating how the porosity represented in a FGM model can be translated into macro-structure in the fabricated part.

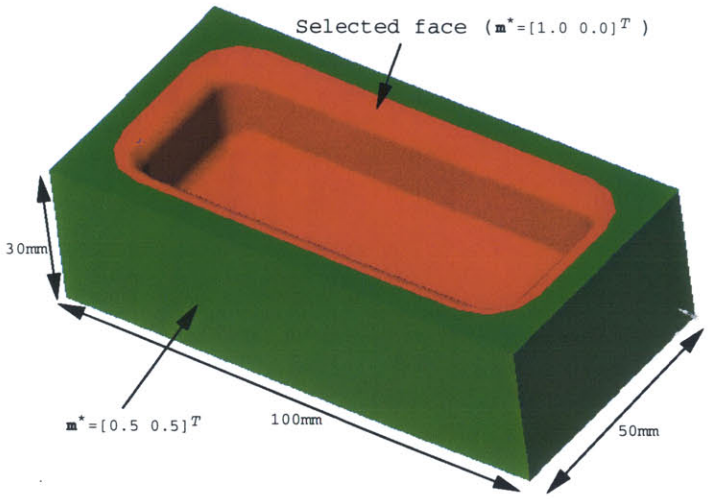
Figure 8-15 illustrates how the composition grading within a model might be designed. Figure 8-15(a) shows the highlighted surface of the cavity boundary and the desired grading of the density of the material as a function of distance from this feature. The intended grading ($\mathbf{m}^*(\mathbf{x})$) is a quadratic function of distance, r, smoothly blending the fully dense material at the cavity boundary into the block interior with 50% porosity:

$$\mathbf{m}^*(\mathbf{x}) = \mathbf{m}(r) = \begin{cases} \begin{bmatrix} 1 - \frac{1}{3}r + \frac{1}{18}r^2 \\ \frac{1}{3}r - \frac{1}{18}r^2 \end{bmatrix} & \text{for } r \leq 3 \\ \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} & \text{otherwise} \end{cases} \tag{8.9}$$

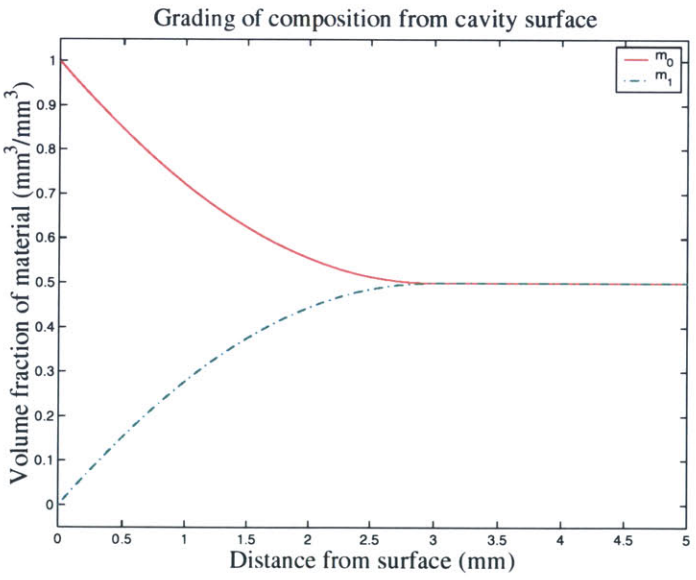
where r is the minimum distance of the point $\mathbf{x}$ from the cavity's boundary, highlighted in Figure 8-15(a).

Three approaches are considered for modeling this part: voxel-based, tetrahedral mesh, and generalized decomposition (Cell-Tuple-Graph and Radial-Edge). The representation of the block as a tetrahedral mesh is illustrated in Figures 8-16 and 8-17 and an exploded view of the model decomposed into FGMDomains is shown in Figure 8-18.

[1] For a more detailed description of halftoning with a dithering array, see Ulichney [70] or Foley *et al* [19].



(a)



(b)

Figure 8-15: (a) Initial compositions of block and the selection of the desired faces from with the composition will be graded. (b) Desired grading from the selected feature.

Voxel-based representation of a block with cavity

For the voxel-based approach, the parameters of the model are substituted into the expression for the storage requirements as defined in Equation 5.21. The minimum feature size of the model (both geometric and material) is the cavity wall's thickness: ($\mu_g = \mu_m = 10\text{mm}$). The maximum rate of change of the desired composition occurs at the cavity's surface along the direction normal to the surface, where

$$M^* = \vec{\nabla} m_0^*(\mathbf{x}) \cdot \hat{\mathbf{v}} = \frac{1}{3}\text{mm}^{-1}$$

This information allows an expression for the memory required to represent the model in a voxel-based representation in terms of the desired geometric and material accuracy.

$$\begin{aligned} S_{vox}[L_x = 100\text{mm}, L_y = 50\text{mm}, & \\ L_z = 30\text{mm}, a, \epsilon_m] &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{1}{4}\left\lceil\frac{100}{a}\right\rceil\left\lceil\frac{50}{a}\right\rceil\left\lceil\frac{30}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil \\ &= 64 + \frac{1}{4}\left\lceil\frac{100}{a}\right\rceil\left\lceil\frac{50}{a}\right\rceil\left\lceil\frac{30}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil + S_{ms} \\ & \qquad \text{where } a = \min\{\epsilon_g, 10, 2\sqrt{3}\epsilon_m\} \end{aligned}$$

Tetrahedral mesh

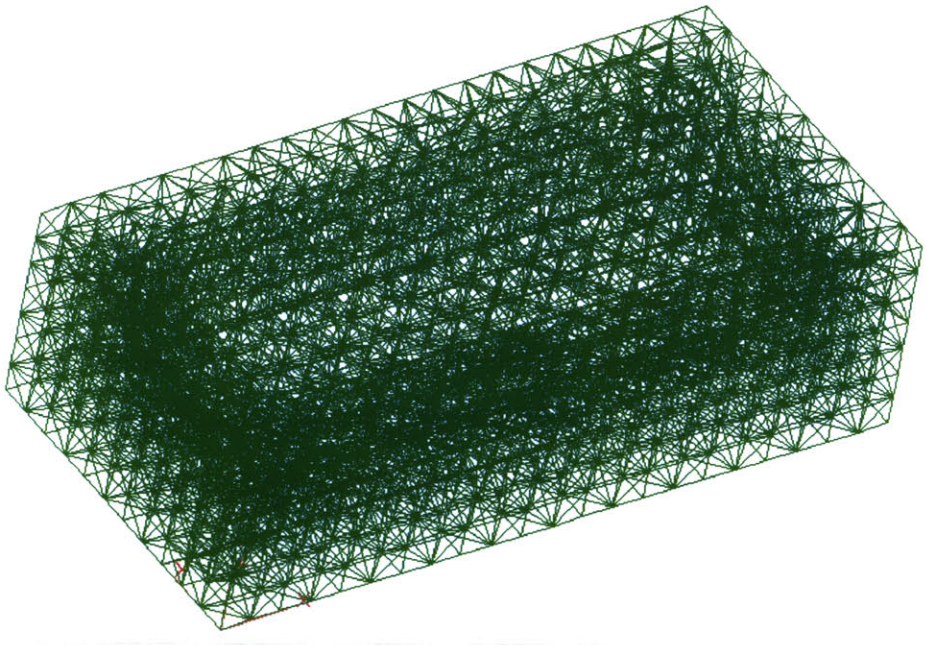


Figure 8-16: Approximation of block geometry with tetrahedra within a finite element mesh. (2206 boundary (external) facets, 8685 tetrahedra, and 2197 nodes)

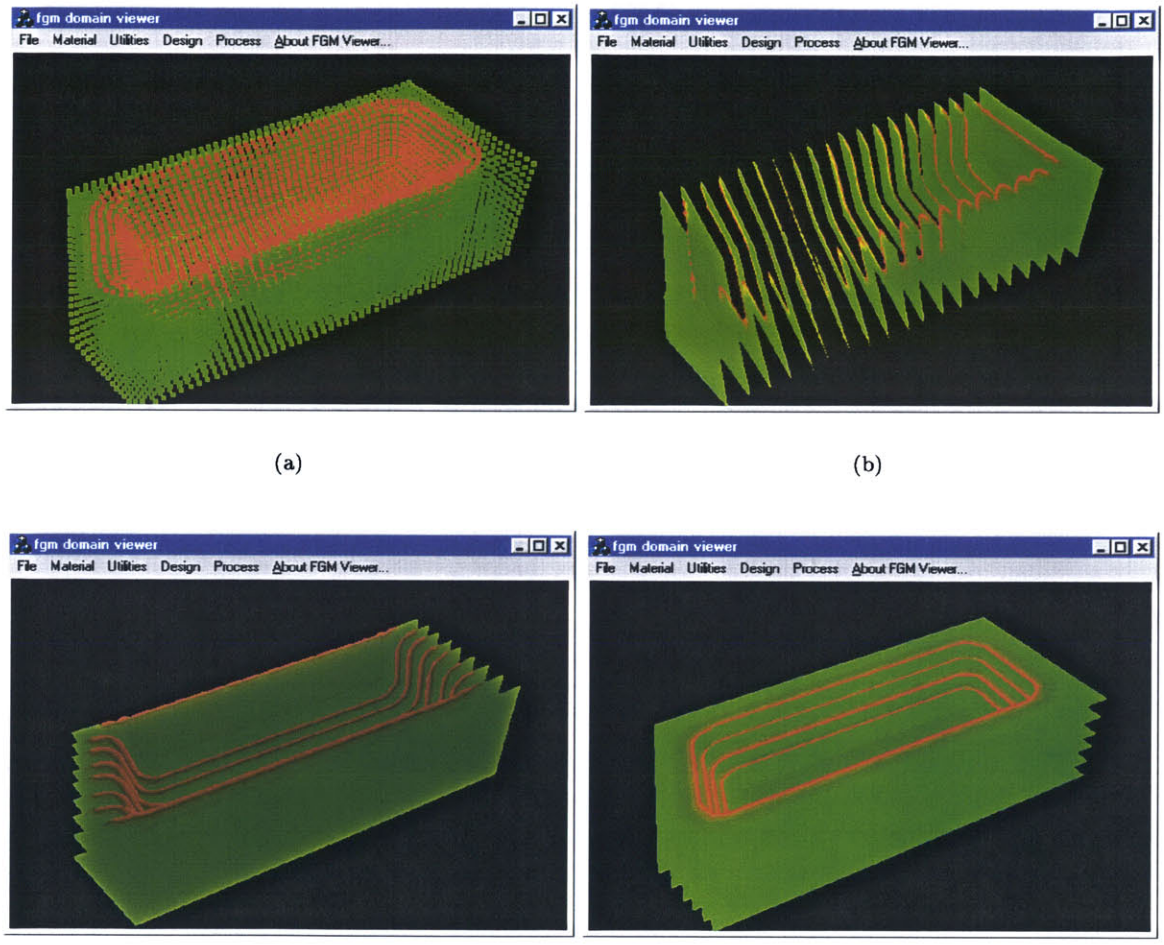


Figure 8-17: (a) Nodes of the tetrahedral mesh colored according to their assigned compositions. View of composition grading assigned to tetrahedral mesh over slices defined by the planes (b) $\pi : x = x_{offset}$, (c) $\pi : y = y_{offset}$ and (d) $\pi : z = z_{offset}$.

The tetrahedral meshed model storage requirements, according to Equation 6.41, is a function of the desired accuracy as well as the model's volume, maximum geometric curvature, and maximum material curvature. An illustration of the block represented as a tetrahedral mesh is given in Figure 8-16. Figure 8-17 shows the composition over the tetrahedral mesh several different ways. The volume of this model is $V_{interior} = 102350\text{mm}^3$. All fillets in the model have a constant radius of curvature of 5mm, therefore $\kappa_{g,max} = \frac{1}{5}\text{mm}^{-1}$. The final factor, the maximum material curvature, needs to be evaluated. Since the grading only occurs along directions normal to the surface, the maximum material curvature will occur along this direction as well.

$$\begin{aligned}
\kappa_{m_0}\Big|_{\hat{\mathbf{n}}_{surface}} &= \frac{\vec{\nabla}[\vec{\nabla}m_0^*(\mathbf{x})\cdot\hat{\mathbf{n}}_{surface}]\cdot\hat{\mathbf{n}}_{surface}}{\left[1+\left(\vec{\nabla}m_0^*(\mathbf{x})\cdot\hat{\mathbf{n}}_{surface}\right)^2\right]^{\frac{3}{2}}} \\
&= \frac{\frac{\partial^2}{\partial r^2}m_0^*(r)}{\left[1+\frac{\partial}{\partial r}m_0^*(r)^2\right]^{\frac{3}{2}}} \\
&= \begin{cases} \frac{81}{(90-6r+r^2)^{\frac{3}{2}}} \text{ for } r \leq 3 \\ 0 \text{ otherwise} \end{cases}
\end{aligned}$$

The material curvature is greatest over the normal offset surface at a distance of $3mm$ from the cavity's boundary, or the interface between the quadratically graded regions and the uniform, porous region. Over this surface, the material curvature is $\kappa_{m,max} = \frac{1}{9}\text{mm}^{-2}$.

Substituting this information into the equation for the number of tetrahedra in the model, an upper bound on the number of tetrahedra necessary to achieve the desired material and geometric accuracy is formed:

$$\begin{aligned}
n_{tetrahedra}\,[V_{interior} = 102350\text{mm}^3, \epsilon_g, \\
\mu_g = 5\text{mm}, \kappa_{g,max} = \frac{1}{5}\text{mm}^{-1}, \\
\mu_{mt} = 5\text{mm}, \kappa_{m,max} = \frac{1}{9}\text{mm}^{-2}] &= O\left[\frac{614100\sqrt{2}}{a^3}\right]
\end{aligned}$$

where $a = \min\left\{5\sqrt{3}\arcsin\sqrt{\frac{\epsilon_g}{5}\left(1-\frac{\epsilon_g}{5}\right)},\ 5,18\arcsin\sqrt{\frac{\epsilon_m}{9}\left(2-\frac{\epsilon_m}{9}\right)}\right\}$.

The expression for bounds on the required number of tetrahedra can be substituted into Equations 6.35- 6.40 to determine bounds on the memory requirements for the model within the variations

on the tetrahedral modeling method, summarized as follows:

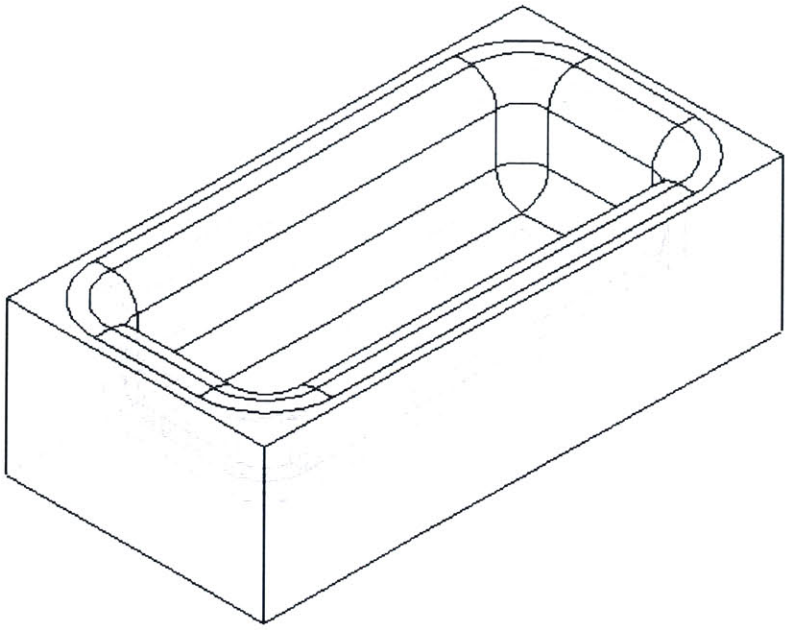
$$\begin{aligned} S_{te_{00}} &\leq 56n_{tetrahedra} + 128 + S_{ms} \\ S_{te_{01}} &\leq 104n_{tetrahedra} + 80 + S_{ms} \\ S_{te_{10}} &\leq 76n_{tetrahedra} + 140 + S_{ms} \\ S_{te_{11}} &\leq 124n_{tetrahedra} + 92 + S_{ms} \end{aligned}$$

Generalized decomposition

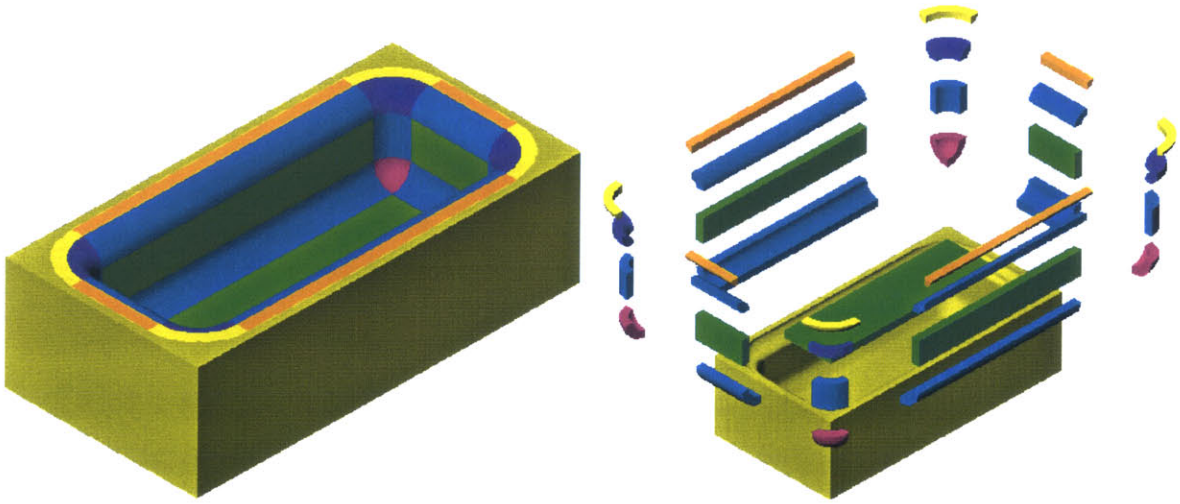
The generalized methods for representing the block with a cavity are capable of representing the desired geometry and composition exactly. To accomplish this, the model is decomposed into FGM-Domains, as shown in Figure 8-18(a) and (b). The numbers of each FGMDomain and the associated storage cost are listed in Table 8.6. By choosing to use quadratic rational pentahedral and hexahedral FGMDomains, both the curved geometry and nonlinear composition are represented exactly. To maintain the adjacency relationship between all of the FGMDomains, a generalized data structure is used. Table 8.6 lists the storage requirements for maintaining relationships between the FGMDomains within the Cell-Tuple-Graph data structure. The corresponding cost for representing the topology within the Radial-Edge database is given in Table 8.7.

Observations

The storage costs for the approaches to modeling the block with a cavity are graphed in Figure 8-19 and 8-20. Unlike the storage costs for modeling the sphere and bar, which were functions of only geometric or material accuracy, respectively), the storage costs for modeling this object are functions of both accuracies. To observe the relationships between memory requirements and geometric accuracy, the storage costs are plotted as functions of geometric accuracy for four different material accuracies in Figure 8-19. Likewise, Figure 8-20 illustrates the variation of the storage costs as functions of material accuracy for four different geometric accuracies. In each case, the storage costs for the Cell-Tuple-Graph and the Radial-Edge methods are constant since they represent the intended geometry and material composition exactly. The requirements for the approximation methods (voxel-based and tetrahedral), however, grow with increasing accuracy. In addition, each graph may contain a break point for each approximation method at which the storage cost transitions from being a function of the corresponding accuracy to being independent of that parameter. This is due to the factor that only one of the following four parameters are used in determining the dimensions of the voxels or the size of the tetrahedra: geometric accuracy, material accuracy, minimum geometric feature size, or minimum material feature size. Consider Figure 8-19(b). For $\epsilon_g < 0.011$mm, the intended geometric accuracy is the dominant factor in determining the dimensions of the voxels. For



(a)



(b) (c)

Figure 8-18: (a) Wireframe view of block decomposed into FGMDomains. (b) View of block with FGMDomains colored according to class and degree of shape and material variation. (c) Exploded view of three dimensional FGMDomains, colored according to their degrees of geometric and material variation.

Classes	*instances*	$\frac{ctgs}{class}$	$\frac{cells}{class}$	$\frac{tuples}{class}$	$\frac{Storage(bytes)}{class}$
FGMPoint	80		80		$400S_{flt}$
FGMRationalBézierCrv					
$n_x = 1, n_m = 0$	80		80		$160S_{int} + 880S_{flt}$
$n_x = 2, n_m = 0$	68		68		$136S_{int} + 1020S_{flt}$
$n_x = 1, n_m = 2$	32		32		$64S_{int} + 544S_{flt}$
FGMRationalBézierTri					
$n_x = 2, n_m = 0$	12		12	144	$24S_{int} + 444S_{flt}$
$n_x = 2, n_m = 2$	8		8	96	$16S_{int} + 256S_{flt}$
FGMRationalBézierQuad					
$(m_x, n_x) = (1,1),\ (m_m, n_m) = (0,0)$	23		23	368	$92S_{int} + 437S_{flt}$
$(m_x, n_x) = (1,1),\ (m_m, n_m) = (2,0)$	24		24	384	$96S_{int} + 600S_{flt}$
$(m_x, n_x) = (2,1),\ (m_m, n_m) = (0,0)$	28		28	448	$112S_{int} + 756S_{flt}$
$(m_x, n_x) = (2,2),\ (m_m, n_m) = (0,0)$	12		12	192	$48S_{int} + 468S_{flt}$
$(m_x, n_x) = (2,1),\ (m_m, n_m) = (0,2)$	28		28	448	$112S_{int} + 924S_{flt}$
FGMRationalBézierPent					
$(m_x, n_x) = (1,2),\ (m_m, n_m) = (2,0)$	4		4		$16S_{int} + 210S_{flt}$
$(m_x, n_x) = (2,2),\ (m_m, n_m) = (0,2)$	4		4		$16S_{int} + 324S_{flt}$
$(m_x, n_x) = (2,2),\ (m_m, n_m) = (2,0)$	4		4		$16S_{int} + 306S_{flt}$
FGMRationalBézierHex					
$(l_x, m_x, n_x) = (1,1,1),$					
$(l_m, m_m, n_m) = (0,0,2)$	5		5		$30S_{int} + 205S_{flt}$
$(l_x, m_x, n_x) = (1,1,2),$					
$(l_m, m_m, n_m) = (0,0,2)$	12		12		$72S_{int} + 684S_{flt}$
$(l_x, m_x, n_x) = (1,2,2),$					
$(l_m, m_m, n_m) = (0,0,2)$	4		4		$24S_{int} + 468S_{flt}$
FGMBRepRegion	2	1	2		$4S_{flt} + 2S_{ptr}$
CellTupleGraph		1			$2S_{ptr}$
Cell			430		$860S_{int} + 860S_{ptr} + S_{ms}$
Tuple				2080	$2080S_{int} + 18720S_{ptr}$
TOTAL					$3974S_{int} + 8930S_{flt} + 19584S_{ptr} + S_{ms}$
					165672 bytes + S_{ms}

Table 8.6: FGMDomain and Cell-Tuple-Graph objects required to represent FGM block-with-cavity object exactly and the associated storage cost.

FGMDomains	$1034S_{int} + 8930S_{flt} + 2S_{ptr}$	
Classes	*instances*	$\frac{Storage(bytes)}{class}$
Complex	1	$S_{ptr} + S_{ms}$
Region	35	$175S_{ptr}$
Shell	35	$140S_{ptr}$
Face	135	$270S_{ptr}$
Loop	135	$135S_{ptr}$
Edge	180	$360S_{ptr}$
Vertex	80	$160S_{ptr}$
FaceUse	270	$1620S_{ptr}$
LoopUse	270	$1620S_{ptr}$
EdgeUse	1064	$7448S_{ptr}$
VertexUse	1064	$4256S_{ptr}$
TOTAL	$1034S_{int} + 8930S_{flt} + 16187S_{ptr} + S_{ms}$	
	140324 bytes + S_{ms}	

Table 8.7: Radial-Edge objects required to represent FGM block-with-cavity object exactly and the associated storage cost.

$\epsilon_g > 0.011$mm, the material accuracy becomes dominant. Likewise, in Figure 8-19(d), the geometric accuracy determines the size of the tetrahedra for $\epsilon_g < 0.009$mm, while the material accuracy becomes the limiting factor for $\epsilon_g > 0.009$mm. A similar explanation holds true for the graphs in Figure 8-20 in which the storage requirements are plotted versus the desired material accuracy. In these cases, the breakpoints occur where the material accuracy constraint is relaxed to the point that it no longer dominants and some other factor (usually geometric accuracy) limits the dimensions of the voxels or tetrahedra.

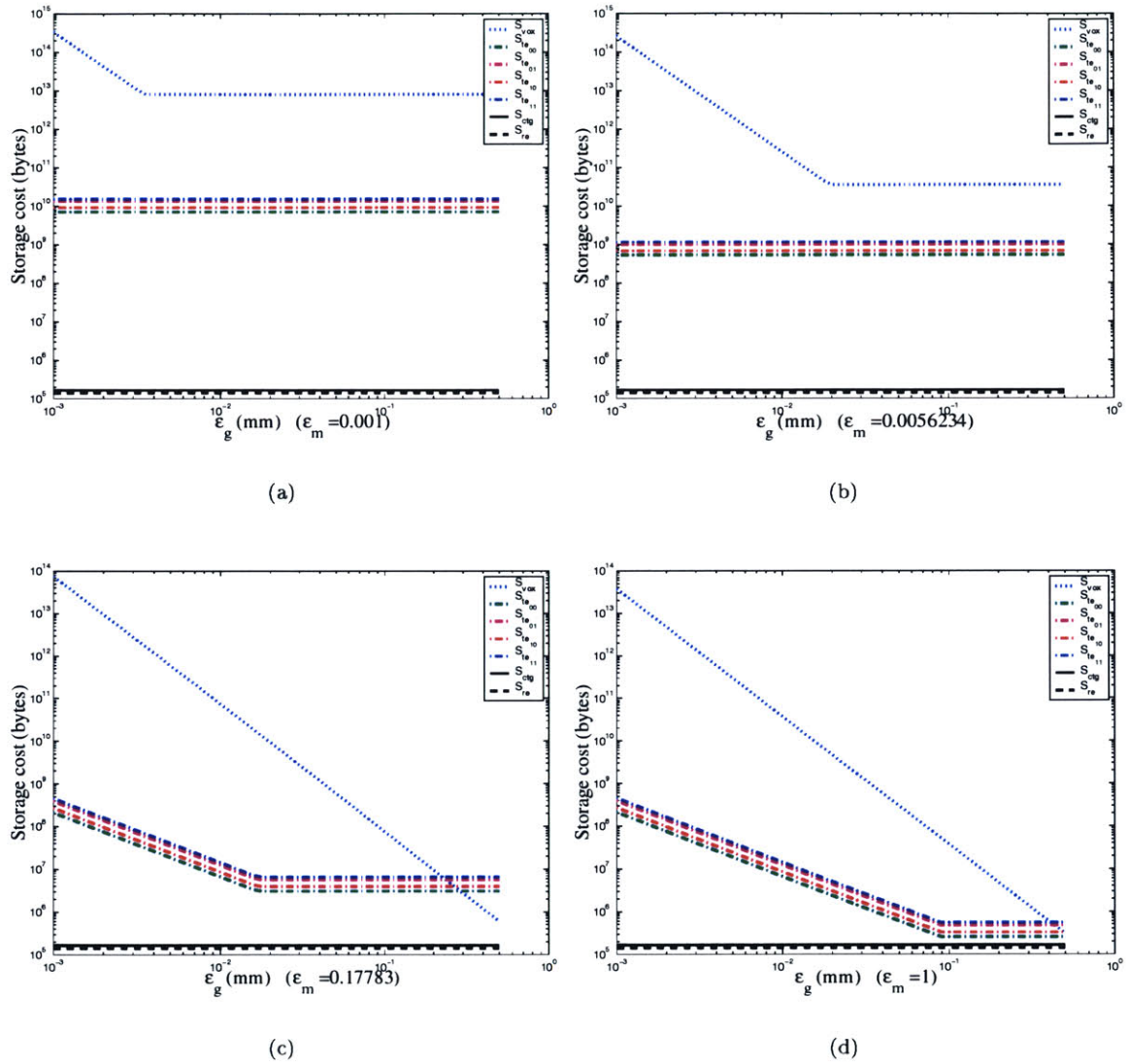


Figure 8-19: Graph of storage cost for representing a block (with composition graded from the boundary of a cavity) as functions of geometric accuracy (a) $\epsilon_m = 0.001$, (b) $\epsilon_m = 0.0056234$, (c) $\epsilon_m = 0.17783$, and (d) $\epsilon_m = 1$.

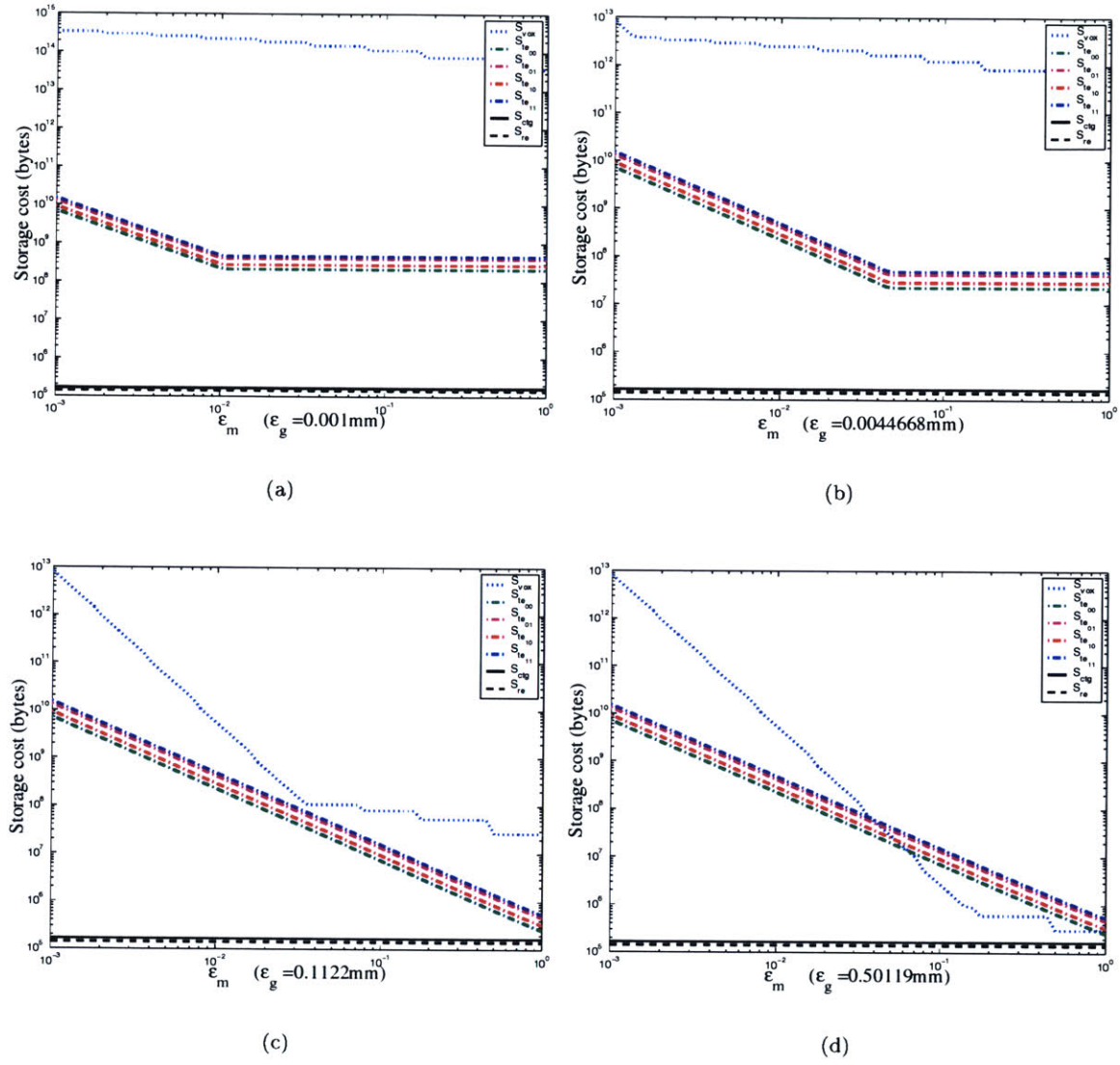


Figure 8-20: Graph of storage cost for representing a block (with composition graded from the boundary of a cavity) as functions of material accuracy (a) $\epsilon_g = 0.001$mm, (b) $\epsilon_g = 0.00446688$mm, (c) $\epsilon_g = 0.1122$mm, and (d) $\epsilon_g = 0.50119$mm

8.2.4 Cylinder butted to plate

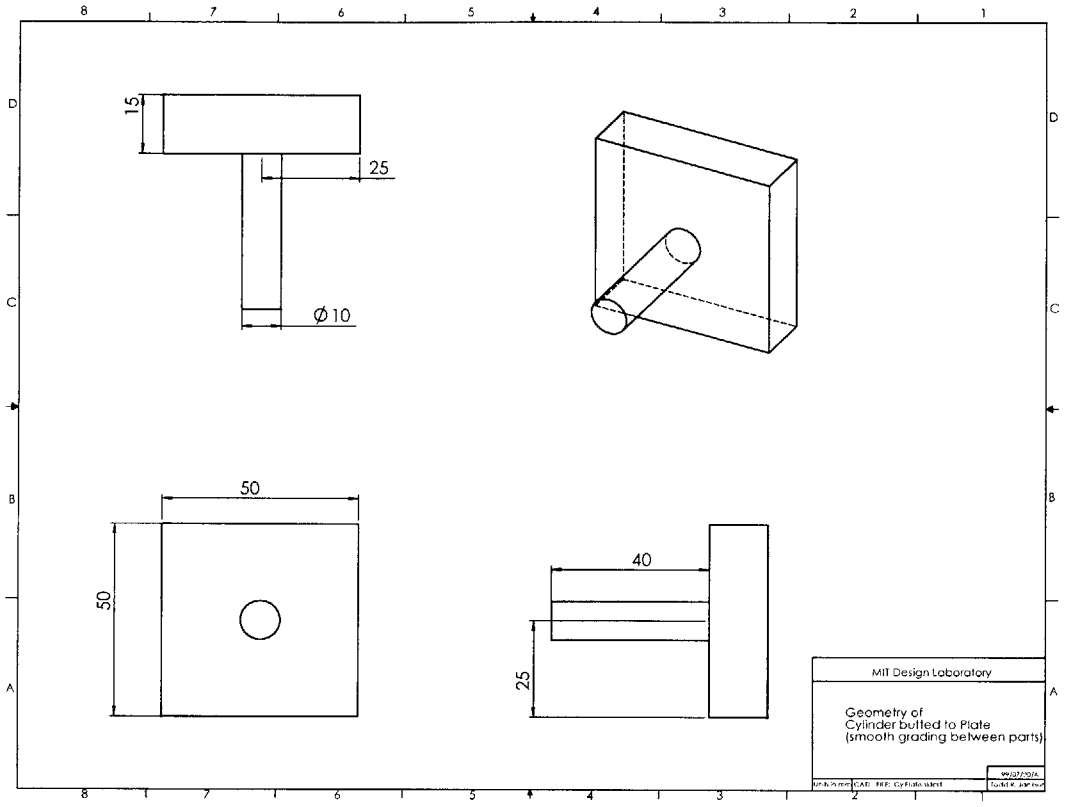


Figure 8-21: Geometric design of cylinder butted to plate.

One of the proposed design methods in Chapter 7 is the use of material blends. To illustrate how such a design tool could be used and its impact the memory requirements, the design of the composition in a model of a cylinder butted to a plate is studied. To begin the design, the geometric design of a cylinder butted to the plate is created, as shown in Figure 8-21. The cylinder is uniformly assigned one material (m_0) and the plate is uniformly assigned a second, different material (m_1). This requires the definition of a three dimensional Material Space (the third material being voids). Across the interface between the cylinder and plate, the composition initially is discontinuous. The initial, piece-wise constant compositions are illustrated in Figure 8-22.

For the design of a blend region, the face incident to both of the regions would be selected as the surface to be "filleted" and a distance over which the blend is to be performed is specified, as shown in Figure 8-23(a). For this case, the distance of 10mm is given. From the existing geometry and compositions, the model is modified so as to contain a smoothly graded region between the piece-wise constant compositions in the plate and the cylinder. Figure 8-23(b) shows the material filleted section between the two constant regions.

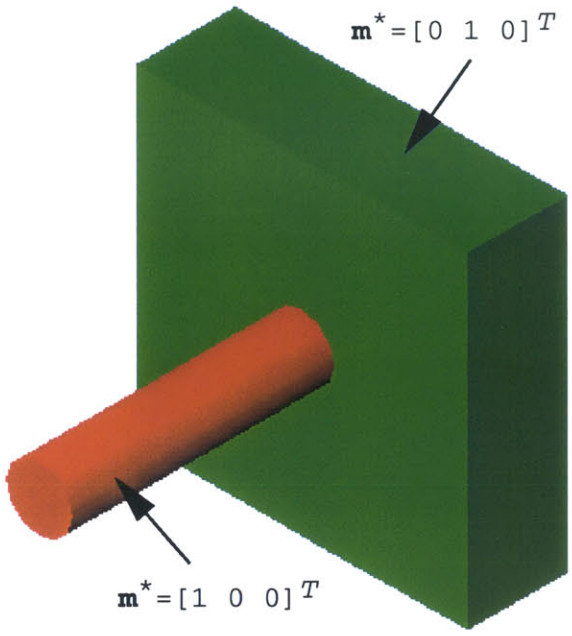


Figure 8-22: Initial, piece-wise constant compositions assigned to cylinder and plate.

$$\mathbf{m}^*(\mathbf{x}) = \begin{cases} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ cylinder and } r > 10, \\ \begin{bmatrix} \frac{1}{2} + \frac{3}{40}r - \frac{1}{4000}r^3 \\ \frac{1}{2} - \frac{3}{40}r + \frac{1}{4000}r^3 \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ cylinder and } r \leq 10, \\ \begin{bmatrix} \frac{1}{2} - \frac{3}{40}r + \frac{1}{4000}r^3 \\ \frac{1}{2} + \frac{3}{40}r - \frac{1}{4000}r^3 \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ plate and } r \leq 10, \\ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ plate and } r > 10, \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \text{otherwise} \end{cases} \tag{8.10}$$

where r is the unsigned minimum distance from $\mathbf{x}$ to the interface.

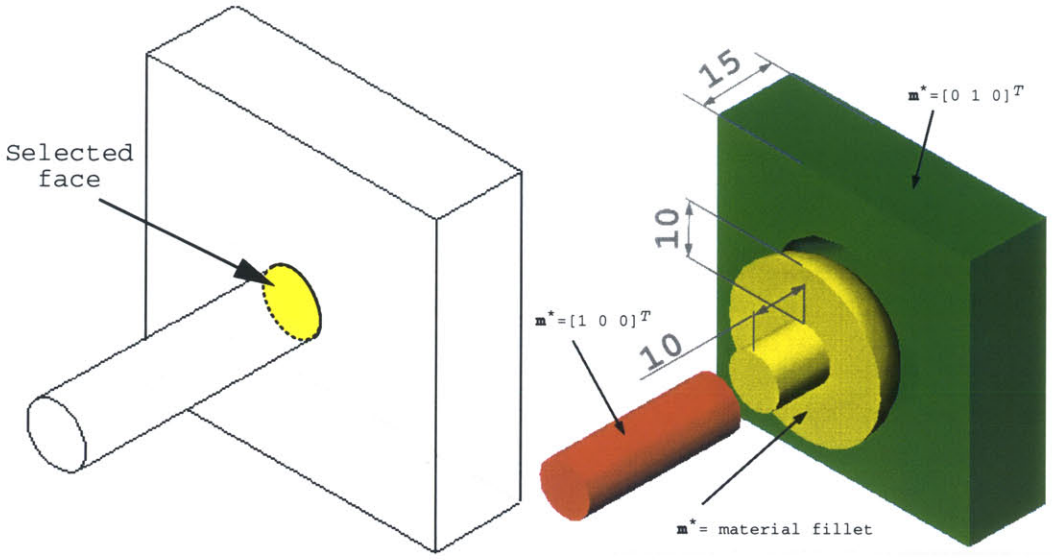


Figure 8-23: (a) Selection of the desired face across which composition will be filleted. (b) Decomposition of model into desired piece-wise constant regions and material fillet.

The preceding scenario describes how a designer might hypothetically design an FGM part using material blends. To represent the initial and final models, however, a solid modeling representation must be chosen. Three methods are analyzed here in terms of their memory requirements. The representations of the model by the different approaches are shown in Figures 8-24, 8-25, and 8-26, including the exhaustive enumeration of the object as voxels, a tetrahedral mesh, and the generalized decomposition of the cylinder and plate into FGMDomains.

Voxel-based representation of cylinder butted to a plate

The storage requirement for a voxel-based model of the cylinder and plate (Figure 8-24) is initially a function of the dimensions of the model (L_x = 50mm, L_y = 50mm, and L_z = 55mm), feature sizes (μ_g = 10mm and μ_m =10mm), the desired geometric and material accuracies (ϵ_g and ϵ_m), before the material filleting operation is performed. After material filleting, the maximum gradient of the composition must also be considered. The point of maximum change in composition occurs over the interface surface in the direction normal to the surface, where $M^* = \frac{3}{40}\text{mm}^{-1}$. Referring

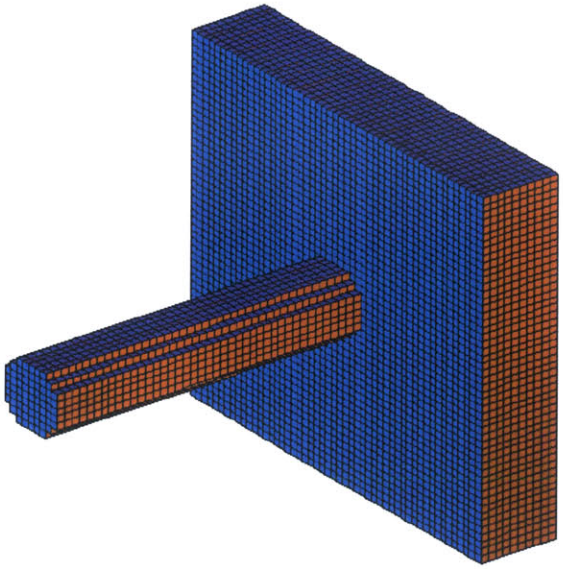


Figure 8-24: Voxel-based representation of cylinder and plate.

to Equation 5.21, the storage cost for this model in terms of these parameters is:

$$
\begin{aligned}
S_{vox}[L_x = 50\text{mm}, L_y = 50\text{mm},& \\
L_z = 55\text{mm}, a, \epsilon_m] &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{3}{8}\left\lceil\frac{50}{a}\right\rceil^2\left\lceil\frac{55}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil \\
&= 64 + \frac{3}{8}\left\lceil\frac{50}{a}\right\rceil^2\left\lceil\frac{55}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil + S_{ms} \text{ bytes} \qquad (8.11) \\
&\quad \text{where } a = \min\{\epsilon_g, 10, \frac{80\sqrt{3}\epsilon_m}{9}\}
\end{aligned}
$$

Tetrahedral mesh

Figure 8-25 is a wireframe view of the model decomposed into tetrahedra. The storage cost for this representation depends on the volume of the model as well as the desired accuracy in representing the geometry and the material composition. Before filleting, only the geometric curvature must be considered. After filleting, the material curvature must also be considered to capture the composition intent of the smoothly graded interface region. The maximum material curvature occurs at an offset distance of 10 mm from the interface surface between the cylinder and the plate, where the material curvature is $\frac{3}{200}$ mm^{-1}. For all four tetrahedral modeling methods presented in Section 4.5, the storage cost is a function of the maximum number of tetrahedra needed to accurately convey this

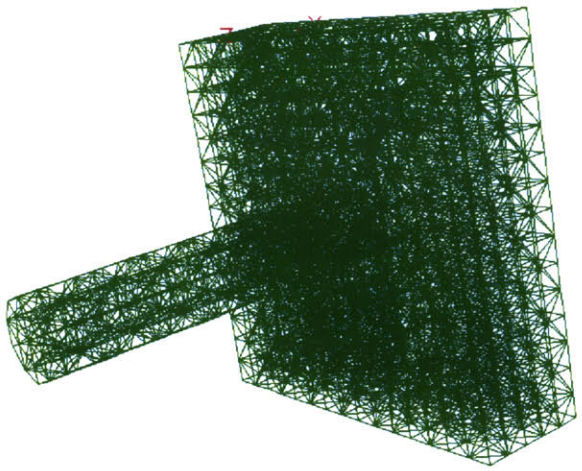


Figure 8-25: Tetrahedral mesh model of cylinder and plate (1708 boundary facets, 7985 tetrahedra, and 1970 nodes).

information.

$$n_{tetrahedra}[V_{interior} = 40624\text{mm}^3, \epsilon_g, \mu_g = 10\text{mm}, \kappa_{g,max} = \frac{1}{5}\text{mm}^{-1}, \mu_{mt} = 10\text{mm}, \kappa_{m,max} = \frac{3}{200}\text{mm}^{-2}] = O\left[\frac{243850\sqrt{2}}{a^3}\right]$$

$$\text{where } a = \min\left\{5\sqrt{3}\arcsin\sqrt{\frac{\epsilon_g}{5}\left(1-\frac{\epsilon_g}{5}\right)},\ 10,\ \frac{400\arcsin\sqrt{\frac{3\epsilon_m}{200}\left(2-\frac{3\epsilon_m}{200}\right)}}{3}\right\}.$$

This bound for the number of tetrahedra can be substituted into Equations 6.35 to 6.40 to determine a bound for the storage costs for the various tetrahedral modeling schemes, as below:

$$S_{te_{00}} = 64n_{tetrahedra} + 152 + S_{ms} \quad (8.12)$$

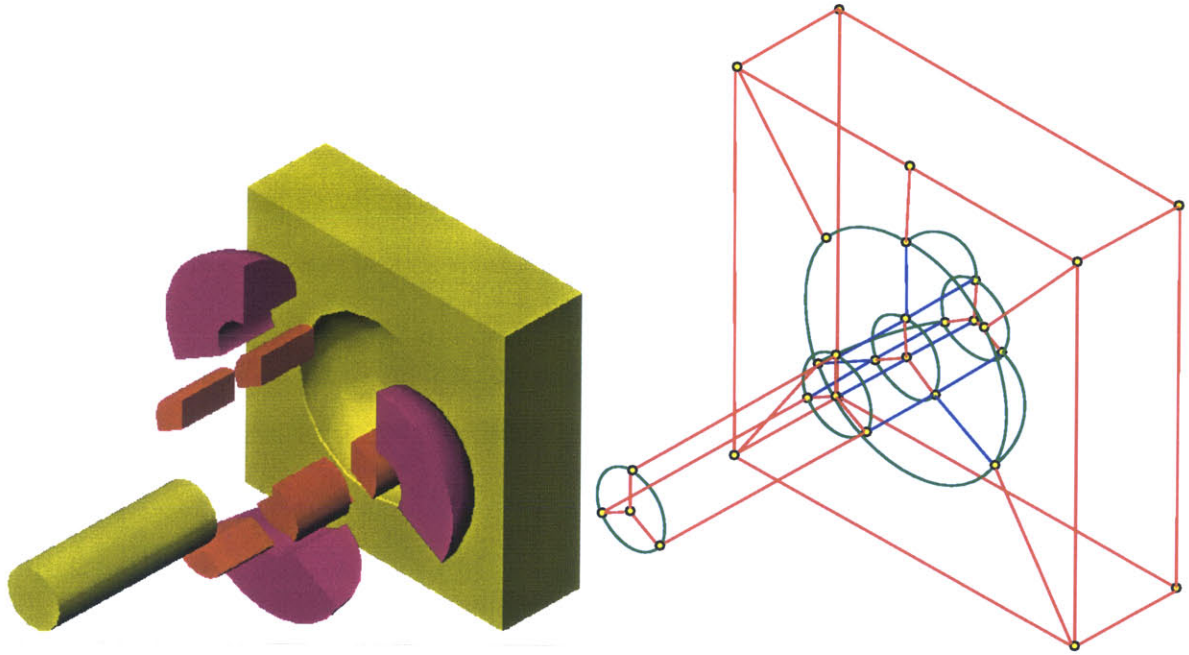
$$S_{te_{01}} = 136n_{tetrahedra} + 80 + S_{ms} \quad (8.13)$$

$$S_{te_{10}} = 84n_{tetrahedra} + 164 + S_{ms} \quad (8.14)$$

$$S_{te_{11}} = 156n_{tetrahedra} + 92 + S_{ms} \quad (8.15)$$

Generalized decomposition

For the generalized representations, the model can be exactly decomposed into rational Bézier FGM-Domains. Initially, the model consists of two general regions of constant compositions, bounded by a collection of planar and cylindrical Bézier surfaces. During the material filleting process, the model is decomposed into Bézier regions and additional surfaces, curves, and vertices to exactly represent the desired grading through the interface. The decomposition of the model into these FGMDomains can be seen in Figure 8-26(a) and 8-26(b). The curved geometry is represented exactly by rational



(a) (b)

Figure 8-26: (a) Decomposition of cylinder and plate into regions to represent grading exactly within a generalized data structure. (b) Edges and vertices in generalized representation of bar butted to plate.

quadratic functions while the composition is exactly modeled with cubic blending functions with the FGMDomains. The numbers of instances of each FGMDomain class (and the degrees of the shape and composition variation) are listed in Table 8.8 along with their costs. Also given in Table 8.8 is the total storage requirements for maintaining complete model within the Cell-Tuple-Graph data structure. Table 8.9 lists the cost associated with maintaining the relationships between the same set of FGMDomains within the Radial-Edge data structure. In both representations, the modeling of the intended geometric and material design is exact.

Classes	*instances*	$\frac{ctgs}{class}$	$\frac{cells}{class}$	$\frac{tuples}{class}$	$\frac{Storage(bytes)}{class}$
FGMPoint	30		30		$180S_{flt}$
FGMRationalBézierCrv					
$n_x = 1, n_m = 0$	33		33		$66S_{int} + 396S_{flt}$
$n_x = 1, n_m = 3$	11		11		$22S_{int} + 264S_{flt}$
$n_x = 2, n_m = 0$	20		20		$40S_{int} + 320S_{flt}$
FGMRationalBézierTri					
$n_x = 2, n_m = 0$	12		12	144	$24S_{int} + 336S_{flt}$
$n_x = 2, n_m = 3$	3		3	36	$6S_{int} + 192S_{flt}$
FGMRationalBézierQuad					
$(m_x, n_x) = (1,1),\ (m_m, n_m) = (0,0)$	5		5	80	$20S_{int} + 100S_{flt}$
$(m_x, n_x) = (1,2),\ (m_m, n_m) = (0,0)$	8		8	128	$32S_{int} + 224S_{flt}$
$(m_x, n_x) = (2,2), (m_m, n_m) = (0,0)$	3		3	48	$12S_{int} + 120S_{flt}$
$(m_x, n_x) = (1,1),\ (m_m, n_m) = (3,0)$	6		6	96	$24S_{int} + 192S_{flt}$
$(m_x, n_x) = (1,2),\ (m_m, n_m) = (3,0)$	9		9	144	$36S_{int} + 360S_{flt}$
FGMRationalBézierPent					
$(m_x, n_x) = (1,2),\ (m_m, n_m) = (3,0)$	6		6		$24S_{int} + 384S_{flt}$
$(m_x, n_x) = (2,2),\ (m_m, n_m) = (0,3)$	3		3		$12S_{int} + 336S_{flt}$
FGMBRepRegion	3	1	3		$9S_{flt} + 3S_{ptr}$
CellTupleGraph		1			$2S_{ptr}$
Cell			152		$304S_{int} + 304S_{ptr} + S_{ms}$
Tuple				676	$676S_{int} + 6084S_{ptr}$
TOTAL					$1298S_{int} + 3413S_{flt} + 6393S_{ptr} + S_{ms}$ 58068 bytes + S_{ms}

Table 8.8: FGMDomain and Cell-Tuple-Graph objects required to represent FGM cylinder-plate object exactly and the associated storage cost.

Observations

The storage costs for the various approaches to modeling the cylinder butted to the plate with material filleting are graphed in Figure 8-27 and 8-28. To observe the relationships between memory requirements and geometric accuracy, the storage costs are plotted as functions of geometric accuracy for four different material accuracies in Figure 8-27. Likewise, Figure 8-28 illustrates the variation of the storage costs as functions of material accuracy for four different geometric accuracies. The same trends observed for the preceding models (of a block with a cavity) are observed here. The storage costs for the Cell-Tuple-Graph and the Radial-Edge methods are again constant with respect

Classes	*instances*	$\frac{Storage(bytes)}{class}$
FGMDomains	$318S_{int} + 3413S_{flt} + 3S_{ptr}$	
Complex	1	$S_{ptr} + S_{ms}$
Region	12	$60S_{ptr}$
Shell	12	$48S_{ptr}$
Face	46	$92S_{ptr}$
Loop	46	$46S_{ptr}$
Edge	64	$128S_{ptr}$
Vertex	30	$60S_{ptr}$
FaceUse	92	$552S_{ptr}$
LoopUse	92	$552S_{ptr}$
EdgeUse	338	$2366S_{ptr}$
VertexUse	338	$1352S_{ptr}$
TOTAL	$318S_{int} + 3413S_{flt} + 5260S_{ptr}$	
	49616 bytes	

Table 8.9: Radial-Edge objects required to represent FGM cylinder-plate object exactly and the associated storage cost.

to the desired accuracy as they represent the intended geometry and material composition exactly. The requirements for the approximation methods (voxel-based and tetrahedral) are also found to grow with increasing accuracy. Just as in the preceding example, break points are observed for each of the approximation methods, where the storage cost transitions from being a function of the corresponding accuracy to becoming independent of that parameter. In Figure 8-27(b), for example, the intended geometric accuracy is the dominant factor in determining the dimensions of the voxels for $\epsilon_g < 0.09$mm. For $\epsilon_g > 0.09$mm, the material accuracy becomes dominant. Similarly, in the same graph, the geometric accuracy determines the size of the tetrahedra for $\epsilon_g < 0.003$mm, while the material accuracy becomes the limiting factor for $\epsilon_g > 0.003$mm. A similar explanation holds true for the graphs in Figure 8-28 in which the storage requirements are plotted versus the desired material accuracy. In these cases, the break points occur where the material accuracy constraint is relaxed to the point that it is no longer dominant and some other factor (usually geometric accuracy) limits the dimensions of the voxels or tetrahedra.

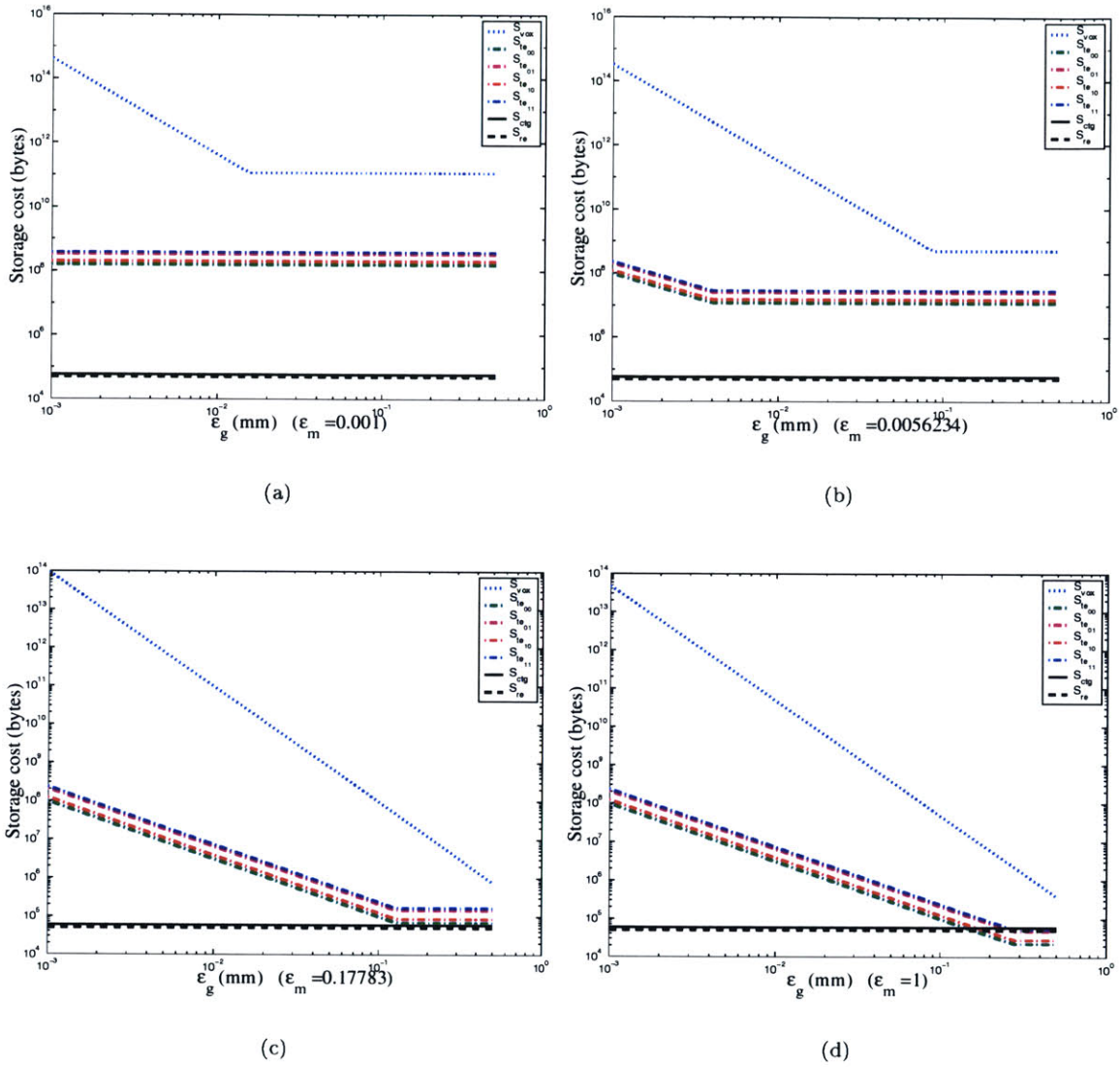


Figure 8-27: Graph of storage cost for representing a cylinder butted to a plate (with a material fillet) as functions of geometric accuracy (a) $\epsilon_m = 0.001$, (b) $\epsilon_m = 0.0056234$, (c) $\epsilon_m = 0.17783$, and (d) $\epsilon_m = 1$.

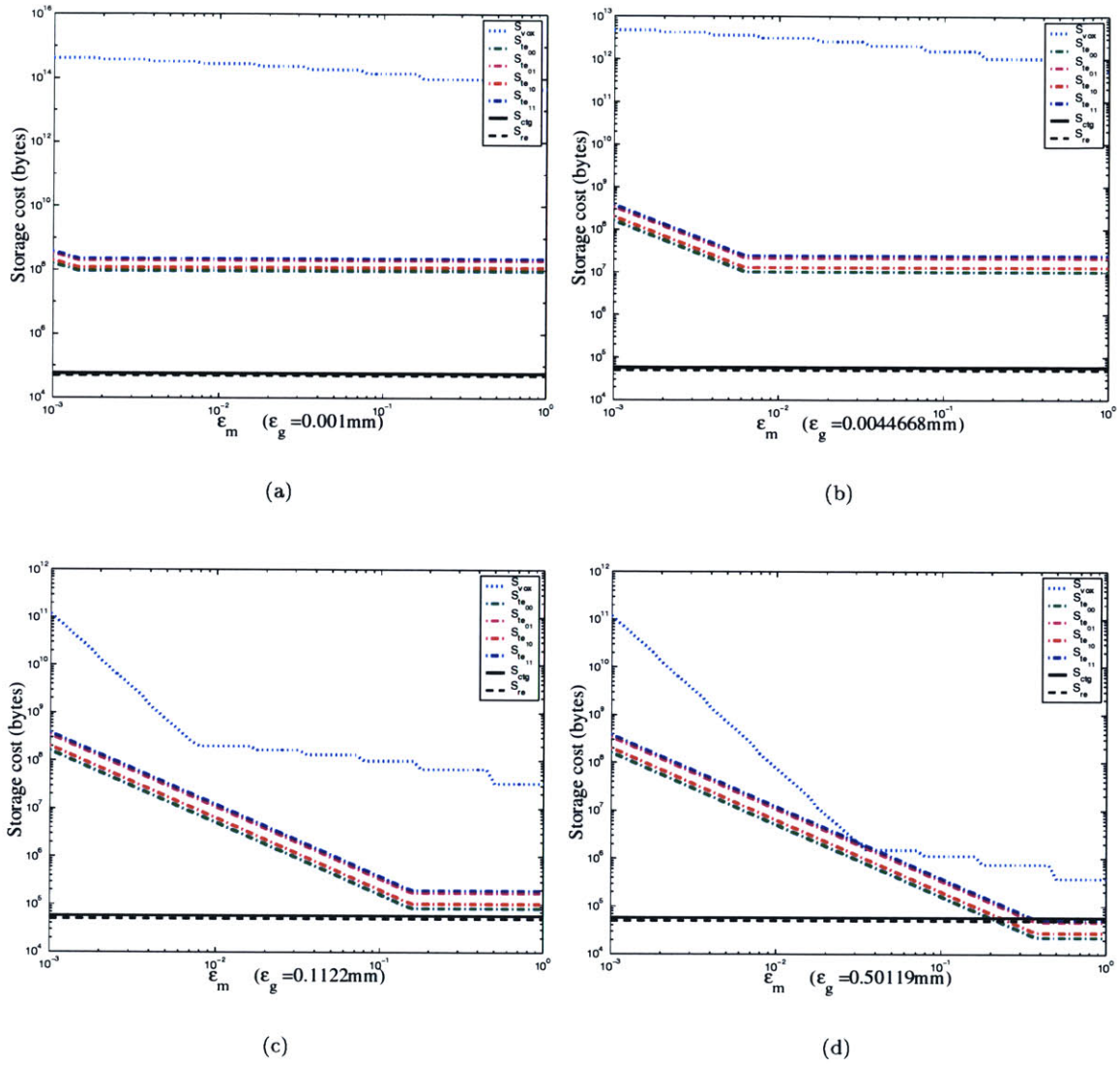


Figure 8-28: Graph of storage cost for representing a cylinder butted to a plate (with a material fillet) as functions of material accuracy (a) $\epsilon_g = 0.001$mm, (b) $\epsilon_g = 0.00446688$mm, (c) $\epsilon_g = 0.1122$mm, and (d) $\epsilon_g = 0.50119$mm.

8.2.5 Drug delivery device

The design of a drug delivery device illustrates the storage cost growth when a model is designed from a library of FGM primitives. In this case, the object is a pill of some base material (m_0) into which models of FGM primitives are placed. For simplicity, the primitives are assumed to consist of equal proportions of the base material (m_0) and some drug (m_1). The size and location of the primitives within the pill affect the profile of the drug release over time, allowing a designer to control the drug release through the FGM design of the device [78].

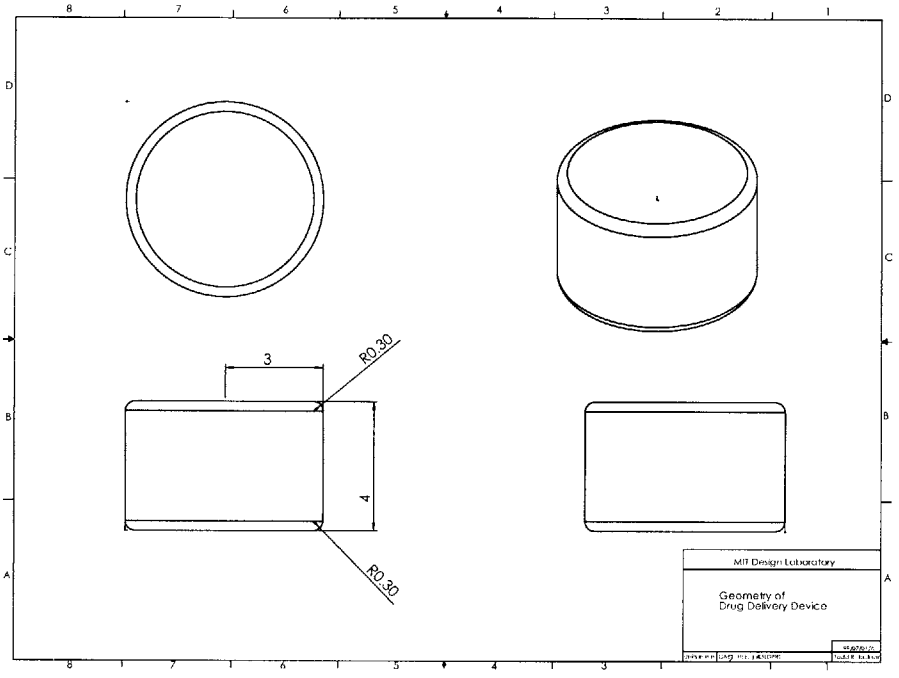


Figure 8-29: Geometric design of boundary of drug delivery device.

The initial design of the pill is illustrated in Figure 8-29 and is assigned a uniform composition. For simplicity, a three dimensional material space is assumed, consisting of the base material or pill material (m_0), the drug (m_1), and void space (m_2).

$$\mathbf{m}^*(\mathbf{x}) = \begin{cases} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ pill }, \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \text{otherwise.} \end{cases} \tag{8.16}$$

Next, the designer makes use of a repository of FGM primitives, positioning copies of the primitives into the pill. A sample library of two primitives is illustrated in Figure 8-30(a), each containing some additional material (m_1) representing the drug. Their placement into the pill is shown in Figure 8-30(b). The composition of the drug delivery device is modified with the introduction of each new primitive, such that:

$$\mathbf{m}^*(\mathbf{x}) = \begin{cases} \begin{bmatrix} m_{0,a}(\vec{\xi}) \\ m_{1,a}(\vec{\xi}) \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ Drug Primitive A,} \\ \begin{bmatrix} m_{0,b}(\vec{\zeta}) \\ m_{1,b}(\vec{\zeta}) \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ Drug Primitive B,} \\ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \text{for } \mathbf{x} \in \text{ pill and not in any Drug Primitive,} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \text{otherwise.} \end{cases} \tag{8.17}$$

where $\vec{\xi}$ and $\vec{\zeta}$ are coordinates local to the primitives A and B.

It should be noted that with each primitive placement, the minimum geometric feature size does not change but the minimum material size may. Since each primitive is being positioned wholly within the boundary of the model, the exterior geometry of the device is unaltered, leaving the minimum geometric feature size equal to the height of the pill ($\mu_g = 4$mm). With each new primitive, however, discontinuities in composition are introduced, potentially reducing the minimum material feature size (μ_m). The minimum material feature size, therefore, is determined by the minimum dimension of all of the primitives as well as the minimum distance between any two primitives or a primitive and the pill's boundary.

The three approaches to FGM modeling are again considered for the drug delivery device. For each case, the relationship between the storage costs and the geometric and material design intent for the pill and each primitive are explored.

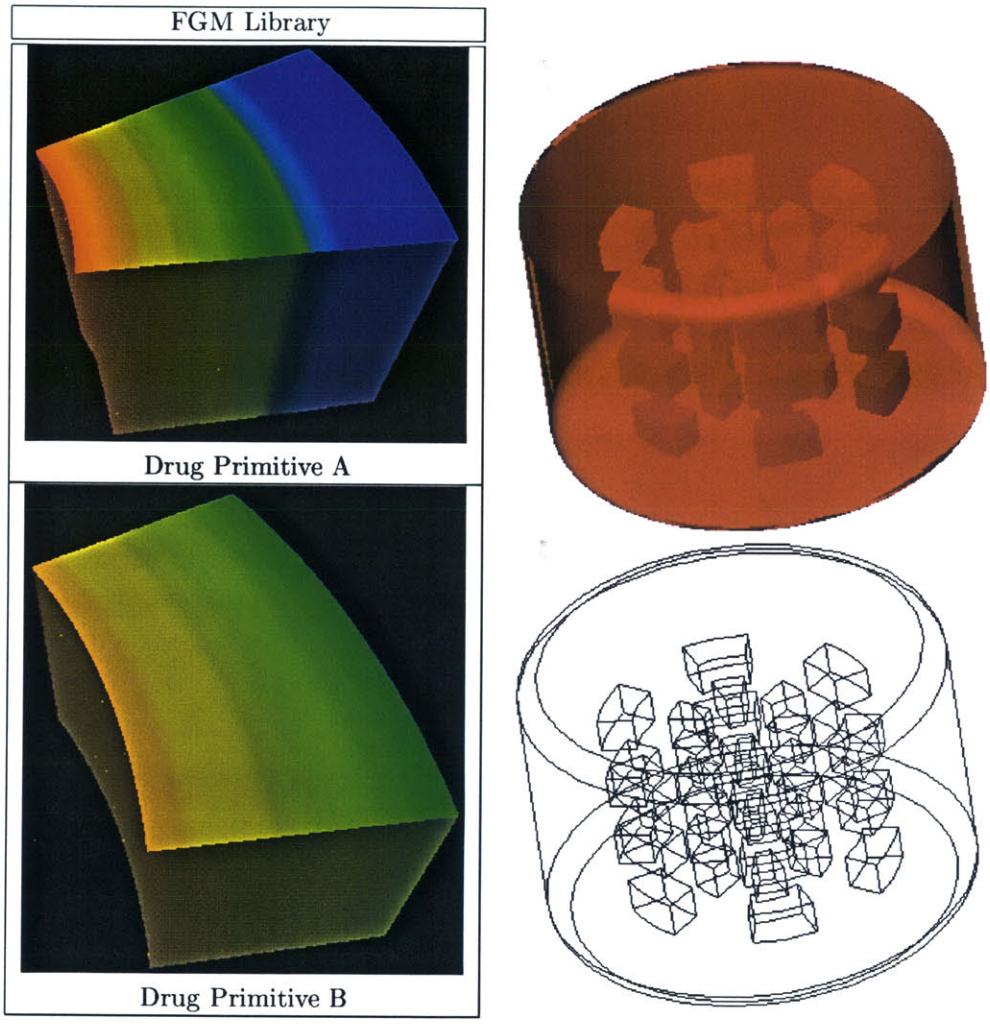


Figure 8-30: (a) Library drug primitives with minimum geometric and material feature sizes μ_g and μ_m. (b) Placement of drug primitives into drug delivery device.

Voxel-based representation of a drug delivery device

The first method to be considered is exhaustive enumeration with voxels of the space occupied by the drug delivery device. Since no restriction is placed on the design of the primitives (except that they be positioned wholly inside the pill), Equation 5.21 of Section 5.6 provides an expression for the storage cost:

$$
\begin{aligned}
S_{vox}[L_x = 6\text{mm}, L_y = 6\text{mm}, & \\
L_z = 4\text{mm}, a, \epsilon_m] &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{3}{8}\left\lceil\frac{6}{a}\right\rceil^2\left\lceil\frac{4}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil \\
&= 64 + \frac{3}{8}d_m\left\lceil\frac{6}{a}\right\rceil^2\left\lceil\frac{4}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil + S_{ms} \qquad (8.18)
\end{aligned}
$$

$$
\text{where } a = \min\{\epsilon_g, \mu_m, \frac{2\sqrt{3}\epsilon_m}{3M^*}\} \text{ and}
$$

$$
M^* = \max\left\{\left|\vec{\nabla}m_0^*(\mathbf{x})\cdot\hat{\mathbf{v}}\right|, \left|\vec{\nabla}m_1^*(\mathbf{x})\cdot\hat{\mathbf{v}}\right|\right\} \qquad (8.19)
$$

Note that the storage cost is no directly dependent of the number of primitives added. The factors determining the memory costs are the dimensions of the pill, the dimensions of the primitives (may limit μ_m), the relative placement of the primitives (may limit μ_m), the geometric and material accuracies, minimum material feature size within a primitive, and the maximum rate of change in composition (M^*) within either primitive. As more features are added, however, the minimum distance between features will gradually decrease as their distribution within the pill becomes increasingly dense, thereby decreasing the minimum material feature size (μ_m) and increasing the storage cost of the voxelized representation.

Tetrahedral mesh

The next class of modeling methods for representing the drug delivery device are the tetrahedral mesh representations, as illustrated in Figure 8-31. For each method, the storage cost is a function of the maximum number of tetrahedra required to accurately capture the geometric and material design intent of both the pill base and the inserted drug primitives. The geometric properties of the pill are readily determined from Figure 8-29 ($\mu_g = 4\text{mm}$, and $\kappa_{g,max} = \frac{10}{3}\text{mm}^{-1}$). Since the primitives may consist of any material distribution in this analysis, the maximum material curvature is left as an unknown ($\kappa_{m,max}$) as is the minimum material feature size (μ_{mt}), which both depend on the nature of the grading within each of the primitives and their spacing from each other and the boundary of the pill.

Depending on the placement of the primitives and their compositions, the limiting parameter is determined and used in evaluating the bound on the number of tetrahedra necessary to represent

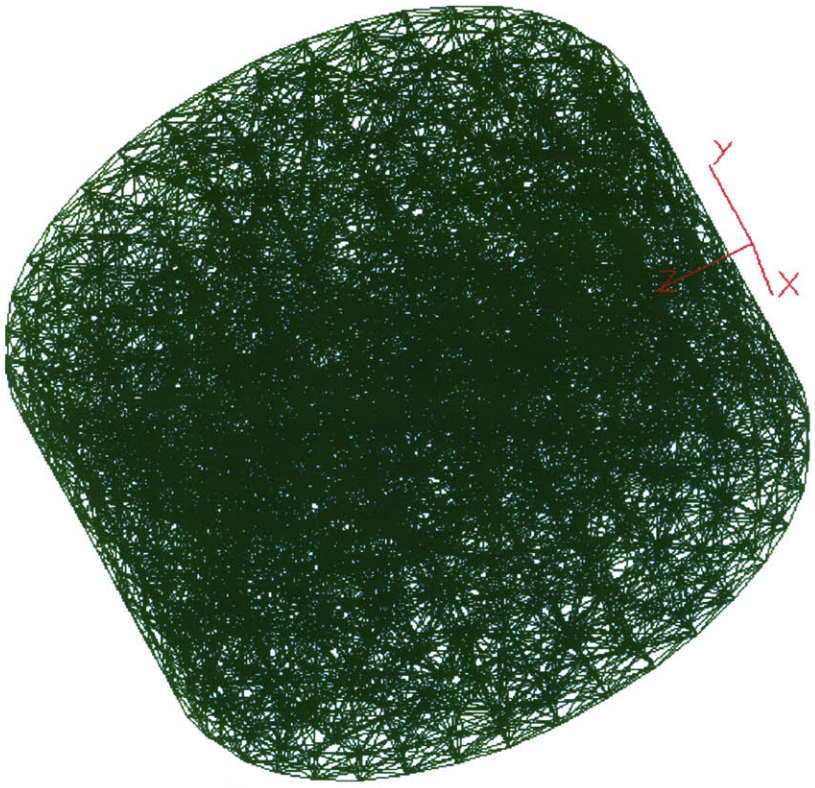


Figure 8-31: Representation of the drug delivery device as a tetrahedral mesh.

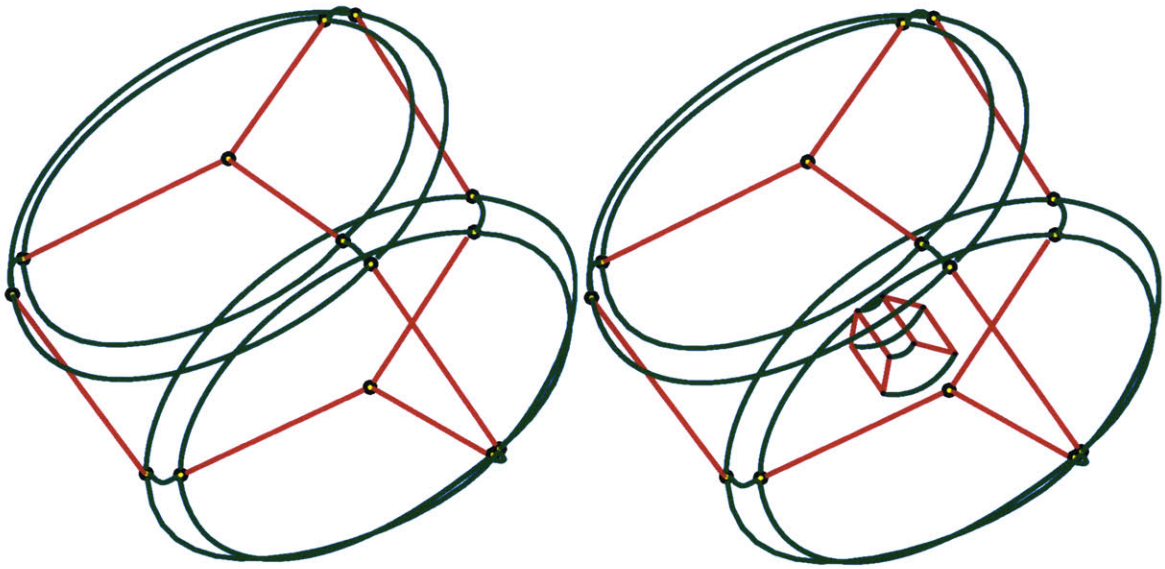
the model, according to Equation 6.41

$$n_{tetrahedra}[V_{interior}, \epsilon_g, \mu_g, \kappa_{g,max}, \mu_{mt}] = O\left[\frac{6\sqrt{2}V_{interior}}{a^3}\right]$$

$$\text{where } a = \min\left\{\frac{3\sqrt{3}}{10}\arcsin\sqrt{\frac{10}{3}\epsilon_g(1-\epsilon_g\frac{10}{3})},\ 4,\right.$$
$$\left.\frac{2}{\kappa_{m,max}}\arcsin\sqrt{\epsilon_m\kappa_{m_i,max}(2-\epsilon_m\kappa_{m,max})}, \mu_{mt}\right\}.$$

The expression for the number of tetrahedra can be inserted into Equations 6.35- 6.40 to determine the bounds for the storage cost for representing the model in terms of a mesh of tetrahedra. Again, the storage cost is not directly dependent on the number of features added to the model. With each feature added, however, the minimum distance between discontinuities in composition (μ_{mt}) may decrease, reducting the allowable size of the tetrahedra and increasing the storage cost required to accurately represent the material composition of the model.

Generalized decomposition



(a) (b)

Figure 8-32: Representation of the drug delivery device as a collection of FGMDomains. Shown are only the vertices and edges bounding the region of uniform composition, into which the drug primitives are placed. (a) Initial vertices and edges of pill. (b) Vertices and edges of device after the addition of a drug primitive.

The generalized modeling schemes represent the pill and the inserted primitives exactly through a collection of FGMDomains stored in a topological database. The decomposition of the pill into

zero and one dimensional rational FGMDomains is illustrated in Figures 8-32(a) and (b). Figure 8-32(a) shows the vertices and edges of the initial design of the pill and Figure 8-32(b) shows the additional vertices and edges after the addition of one primitive. The number and nature of each type of FGMDomain present in the model are listed in Table 8.10. The primitives are each treated as an FGMDomain, although they may in turn be broken into simpler, topological elements.

Each primitive inserted into the model adds additional FGMDomains to the data structure. Since the primitives are considered here to be generic and may be arbitrarily decomposed into additional FGMDomains, the number of primitives (n^A_{dp} and n^B_{dp}) and their storage costs ($n^A_{dp}S^A_{dp}$ and $n^B_{dp}S^B_{dp}$) are listed separately in Table 8.10. The terms n^A_{dpc} and n^A_{dpt} are the number of cells and tuples necessary to represent the topology of Primitive A. Likewise, n^B_{dpc} and n^B_{dpt} serve a similar role for Primitive B.

Classes	*instances*	$\frac{ctg's}{class}$	$\frac{cells}{class}$	$\frac{tuples}{class}$	$\frac{Storage(bytes)}{class}$
FGMPoint	14		14		$84S_{flt}$
FGMRationalBézierCurve $n_x = 1, n_m = 0$	9		9		$18S_{int} + 108S_{flt}$
$n_x = 2, n_m = 0$	18		18		$36S_{int} + 288S_{flt}$
FGMRationalBézier-TriPatch $n_x = 2, n_m = 0$	6		6	72	$12S_{int} + 168S_{flt}$
FGMRationalBézier-QuadPatch $(m_x, n_x) = (1, 2)$, $(m_m, n_m) = (0, 0)$	3		3	48	$12S_{int} + 84S_{flt}$
$(m_x, n_x) = (2, 2)$, $(m_m, n_m) = (0, 0)$	6		6	96	$24S_{int} + 240S_{flt}$
FGMBRepRegion	2	1	2		$2S_{ptr} + 6S_{flt}$
FGMDrugPrimitive A	n^A_{dp}	n^A_{dp}	$n^A_{dp}n^A_{dpc}$	$n^A_{dp}n^A_{dpt}$	$n^A_{dp}S^A_{dp}$
FGMDrugPrimitive B	n^B_{dp}	n^B_{dp}	$n^B_{dp}n^B_{dpc}$	$n^B_{dp}n^B_{dpt}$	$n^B_{dp}S^B_{dp}$
CellTupleGraph	$1 + n^A_{dp} + n^A_{dp}$				$2(1 + n^A_{dp} + n^B_{dp})S_{ptr}$
Cell	$58 + n^A_{dp}n^A_{dpc} + n^B_{dp}n^B_{dpc}$				$\left(58 + n^A_{dp}n^A_{dpc} + n^B_{dp}n^B_{dpc}\right)(2S_{int} + 2S_{ptr}) + \left(n^A_{dp} + n^B_{dp}\right)S_{ptr} + S_{ms}$
Tuple	$216 + n^A_{dp}n^A_{dpt} + n^B_{dp}n^B_{dpt}$				$\left(216 + n^A_{dp}n^A_{dpt} + n^B_{dp}n^B_{dpt}\right)(S_{int} + 9S_{ptr})$
TOTAL	$\left[434 + n^A_{dp}\left(2n^A_{dpc} + n^A_{dpt}\right) + n^B_{dp}\left(2n^B_{dpc} + n^B_{dpt}\right)\right]S_{int} + 978S_{flt} + \left[2064 + n^A_{dp}\left(n^A_{dpc} + 9n^A_{dpt} + 1\right) + n^B_{dp}\left(n^B_{dpc} + 9n^B_{dpt} + 1\right)\right]S_{ptr} + n^A_{dp}S^A_{dp} + n^B_{dp}S^B_{dp} + S_{ms}$				

Table 8.10: Cell-Tuple-Graph objects required to represent FGM drug delivery device object exactly and the associated storage cost.

The Radial-Edge data structure uses instances of the classes listed in Figure 4-9 to maintain the topology of models. For this approach to modeling the drug delivery device, each primitive inserted into the data structure introduces a new shell, referenced by the pill, plus the topological entities and their uses as required to maintain the relationships between the FGMDomains comprising the

Classes	*instances*	$\frac{Storage(bytes)}{class}$
FGMDomains		$102S_{int} + 978S_{flt} + 2S_{ptr}$
Complex	1	S_{ptr}
Region	$2 + n^A_{dp} + n^B_{dp}$	$5\left(2 + n^A_{dp}n^A_{dpr} + n^B_{dp}n^B_{dpr}\right) S_{ptr}$
Shell	$2 + n^A_{dp}n^A_{dps} + n^B_{dp}n^B_{dps}$	$4\left(2 + n^A_{dp}n^A_{dps} + n^B_{dp}n^B_{dps}\right) S_{ptr}$
Face	$15 + n^A_{dp}n^A_{dpf} + n^B_{dp}n^B_{dpf}$	$15\left(2 + n^A_{dp}n^A_{dpf} + n^B_{dp}n^B_{dpf}\right) S_{ptr}$
Loop	$15 + n^A_{dp}n^A_{dpl} + n^B_{dp}n^B_{dpl}$	$15\left(1 + n^A_{dp}n^A_{dpl} + n^B_{dp}n^B_{dpl}\right) S_{ptr}$
Edge	$27 + n^A_{dp}n^A_{dpe} + n^B_{dp}n^B_{dpe}$	$2\left(27 + n^A_{dp}n^A_{dpe} + n^B_{dp}n^B_{dpe}\right) S_{ptr}$
Vertex	$14 + n^A_{dp}n^A_{dpv} + n^B_{dp}n^B_{dpv}$	$2\left(14 + n^A_{dp}n^A_{dpv} + n^B_{dp}n^B_{dpv}\right) S_{ptr}$
FaceUse	$30 + n^A_{dp}n^A_{dpfu} + n^B_{dp}n^B_{dpfu}$	$6\left(30 + n^A_{dp}n^A_{dpfu} + n^B_{dp}n^B_{dpfu}\right) S_{ptr}$
LoopUse	$30 + n^A_{dp}n^A_{dplu} + n^B_{dp}n^B_{dplu}$	$6\left(30 + n^A_{dp}n^A_{dplu} + n^B_{dp}n^B_{dplu}\right) S_{ptr}$
EdgeUse	$108 + n^A_{dp}n^A_{dpeu} + n^B_{dp}n^B_{dpeu}$	$7\left(108 + n^A_{dp}n^A_{dpeu} + n^B_{dp}n^B_{dpeu}\right) S_{ptr}$
VertexUse	$108 + n^A_{dp}n^A_{dpvu} + n^B_{dp}n^B_{dpvu}$	$4\left(108 + n^A_{dp}n^A_{dpvu} + n^B_{dp}n^B_{dpvu}\right) S_{ptr}$
TOTAL		$102S_{int} + 978S_{flt} + 1696S_{ptr} + S_{ms} +$ $n^A_{dp}\left(5n^A_{dpr} + 4n^A_{dps} + 15n^A_{dpf} + 15n^A_{dpl} + 2n^A_{dpe} + 2n^A_{dpv} + 6\,n^A_{dpfu} + 6n^A_{dplu} + 7n^A_{dpeu} + 4n^A_{dpvu}\right) S_{ptr} +$ $n^B_{dp}\left(5n^B_{dpr} + 4n^B_{dps} + 15n^B_{dpf} + 15n^B_{dpl} + 2n^B_{dpe} + 2n^B_{dpv} + 6n^B_{dpfu} + 6n^B_{dplu} + 7n^B_{dpeu} + 4n^B_{dpvu}\right) S_{ptr}$

Table 8.11: Radial-Edge objects required to represent the topology of FGM drug delivery device object exactly and the associated storage cost.

drug primitives. The accounting for the number of instances of each class in the Radial-Edge representation of the pill are given in Table 8.11. The number of drug primitives inserted into the Radial-Edge Data structure are represented by n_{dp}^{A} and n_{dp}^{B}. The numbers of Regions, Shells, Faces, Edges, Vertices, FacesUses, LoopUses, EdgeUses, and VertexUses added to the model with each additional Drug Primitive A are represented by n_{dpr}^{A}, n_{dps}^{A}, n_{dpf}^{A}, n_{dpe}^{A}, n_{dpv}^{A}, n_{dpfu}^{A}, n_{dplu}^{A}, n_{dpeu}^{A}, and n_{dpvu}^{A}, respectively. The numbers of topological entities added with each Drug Primitive B are similarly noted with the superscript 'B' in place of the 'A'.

From this analysis, the storage cost of the drug delivery device in a generalized decomposition modeling method is found to grow with the number of primitives (features) added to the model. Therefore, for each new feature added to a generalized model, the storage cost increases by some constant factor.

Observations

The storage costs for the drug delivery device can not be graphed since the exact material composition of the primitives and their positions are left undefined. This is done intentionally in this analysis to emphasize the different approaches to FGM modeling and how their storage requirements are affected. Expressions for the costs of the voxel-based and tetrahedral data structures depend on the material distribution in the final design, whereas the generalized data structure costs only depend on the number features added to the model.

8.2.6 Widget Mold

The final case to be considered illustrates how the generalized decomposition modeling methods can be extended through the definition of new FGMDomains. The model used as the example is a generic tool, for a "Widget Mold", as shown in Figure 8-33. The composition for the object is designed as a function of distance from the object's boundary by locally controlling the volume fraction of some material (m_1) that increases the hardness of the base material (m_0) [44]. In the analysis here, the generalized decomposition methods incorporate a procedural FGMDomain, illustrating how the introduction of additional FGMDomains to represent the interior of the model can further reduce the memory requirements while maintaining the accuracy of the intended design. Again, the three general approaches to representing the object are considered: voxel-based, tetrahedral mesh, and generalized decomposition.

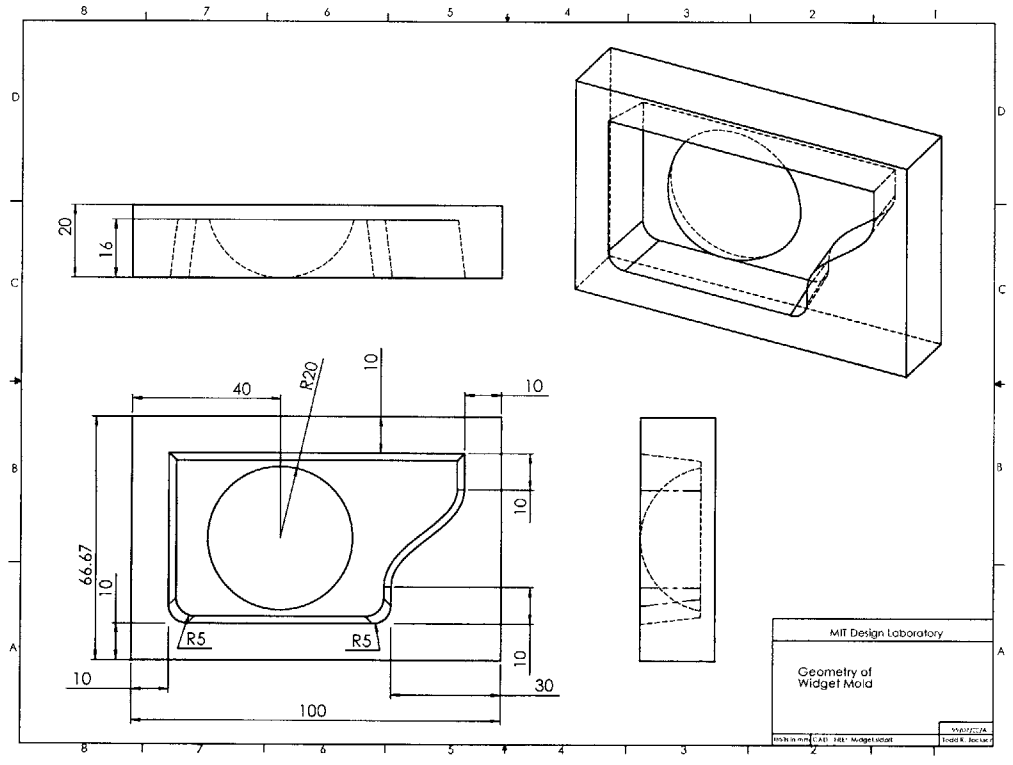


Figure 8-33: Geometric design of Widget Mold.

Given the geometric design of the model in Figure 8-33, the intended composition ($\mathbf{m}^*(\mathbf{x})$) is designed as a function of distance from the object's boundary. In this case, the composition grades linearly over the region within 5mm of the object's boundary, creating a "skin" of linearly varying

FGM:

$$\mathbf{m}^*(\mathbf{x}) = \begin{cases} \begin{bmatrix} 0.7 + \frac{3r}{50} \\ 0.3 - \frac{3r}{50} \\ 0 \end{bmatrix} & \text{for } r \leq 5 \text{ mm and } \mathbf{x} \in \text{ object,} \\ \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} & \text{for } r > 5 \text{ mm and } \mathbf{x} \in \text{ object,} \\ \begin{bmatrix} 0.0 \\ 0.0 \\ 1.0 \end{bmatrix} & \text{otherwise} \end{cases} \tag{8.20}$$

where r is the minimum distance of the point $\mathbf{x}$ from the object's boundary.

Voxel-based representation of the Widget Mold

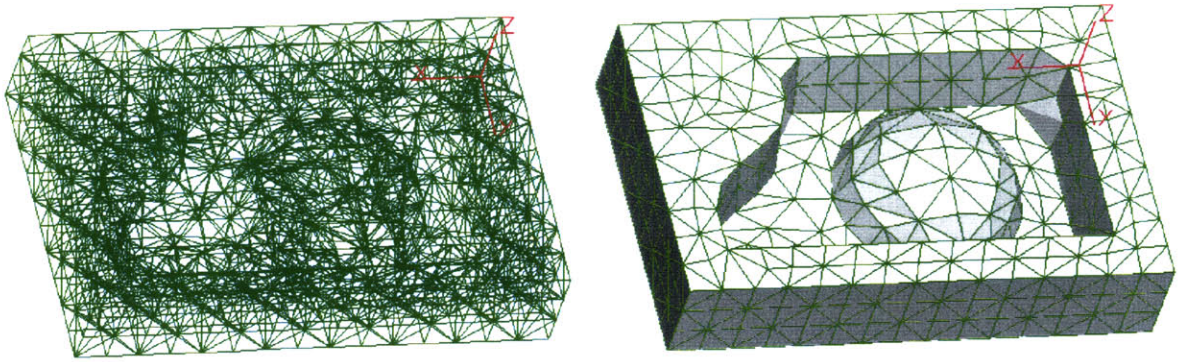
The storage cost for the Widget Mold in a voxel-based representation is again a function of the dimensions of the model, the minimum feature sizes (μ_g and μ_m), and the maximum gradient of material variation.

The minimum geometric feature size of the mold is the thickness of the bottom of the mold: μ_g =4mm. With the linear variation in composition, the minimum material feature size is the distance between two discontinuities in composition. Since the composition varies continuously throughout the model, the minimum material feature size is the same as the minimum geometric feature size: $\mu_m = \mu_g$. Throughout the "skin" of the object, the gradient in composition is constant normal to the surface, yielding a maximum material gradient of $M^* = \frac{3}{50}\text{mm}^{-1}$.

From this information, the expression for the storage cost of the Widget Mold as a voxel-based model is:

$$\begin{aligned} S_{vox}[L_x = 100\text{mm}, & \\ L_y = 66.67\text{mm}, & \\ L_z = 20\text{mm}, a, \epsilon_m] &= 3S_{int} + 6S_{flt} + S_{ptr} + S_{ms} + \frac{3}{8}\left\lceil\frac{100}{a}\right\rceil\left\lceil\frac{66.67}{a}\right\rceil\left\lceil\frac{20}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil \\ &= 64 + \frac{3}{8}\left\lceil\frac{100}{a}\right\rceil\left\lceil\frac{66.67}{a}\right\rceil\left\lceil\frac{20}{a}\right\rceil\left\lceil \lg\left(\frac{1}{2\epsilon_m}+1\right)\right\rceil + S_{ms} \qquad (8.21) \\ & \text{where } a = \min\{\epsilon_g, 4, \frac{100\sqrt{3}\epsilon_m}{9}\} \end{aligned}$$

Tetrahedral mesh representation of the Widget Mold



(a) (b)

Figure 8-34: (a) Representation of the Widget Mold as a tetrahedral mesh (wireframe view). (a) Representation of the Widget Mold as a tetrahedral mesh (solid view).

Figure 8-34 illustrates how the Widget Mold could be represented as a tetrahedral mesh. Figure 8-35 shows the composition values can be assigned to the field nodes in the mesh to define the composition. The memory required to represent this object as a tetrahedral mesh is a function of the number of tetrahedra needed to model the object. Note that the desired composition variation within the model is piece-wise linear, therefore the material curvature is zero and not a factor in determining the size of the tetrahedra. The geometry of the model, however, is curved and the maximum curvature of all of the curves and surfaces occurs at the 5mm fillets. Therefore, $\kappa_{g,max} = 5mm^{-1}$. From voxel-based analysis, the minimum geometric feature is found to be $\mu_g = 4$ mm. In order to capture the grading from the boundary, the minimum material feature must be measured from the mid-plane over the floor of the mold (where a discontinuity in the material derivative exists) to the bottom or the floor of the mold: $\mu_{mt} = 2$mm. With an interior volume of $V_{interior} = 96733\text{mm}^3$, the storage costs for the Widget Mold as a tetrahedral mesh can be determined from Equation 6.41, as

$$
\begin{aligned}
& n_{tetrahedra}[V_{interior} = 96733\text{mm}^3, \epsilon_g, \\
& \mu_g = 4\text{mm}, \kappa_{g,max} = 5\text{mm}^{-1}, \mu_{mt} = 2\text{mm}] \quad = \quad O\left[\frac{580398\sqrt{2}}{a^3}\right] \\
& \qquad\qquad \text{where } a \quad = \quad \min\left\{5\sqrt{3}\arcsin\sqrt{\frac{1}{5}\epsilon_g(1-\frac{1}{5}\epsilon_g)},\ 2\right\}.
\end{aligned}
$$

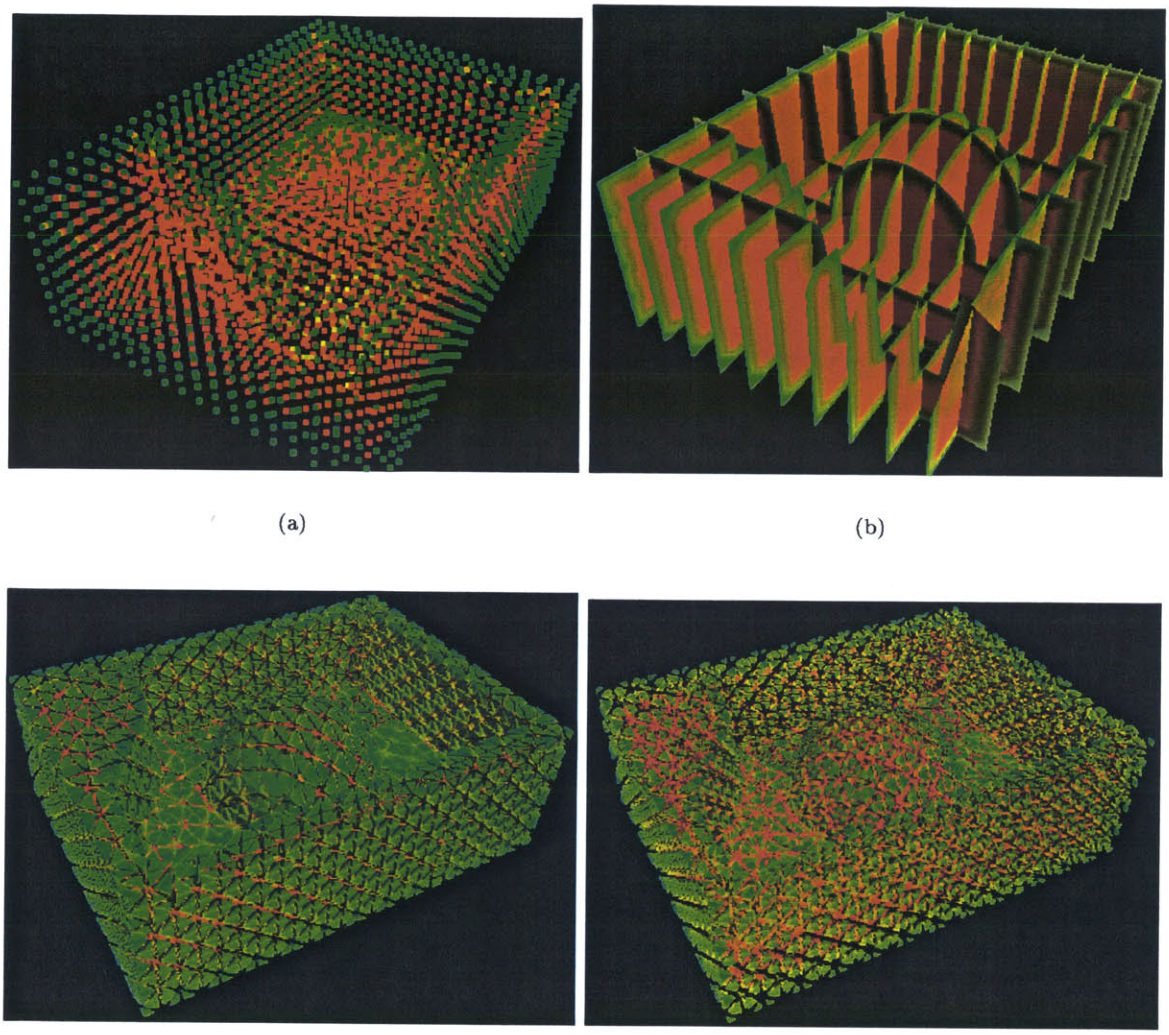
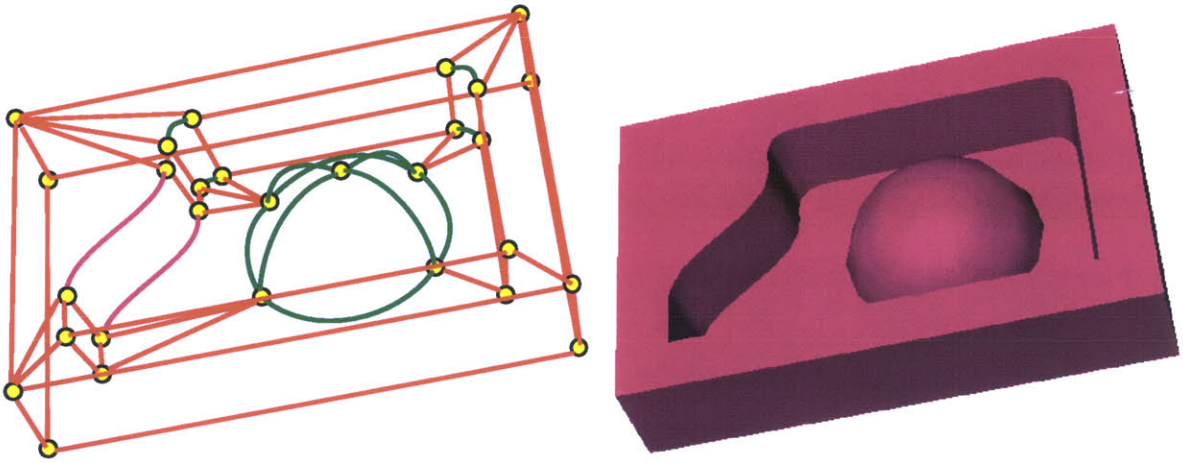


Figure 8-35: (a) View of the nodes in the tetrahedral mesh. (b) View of the material grading over slices of a tetrahedral mesh representation of the Widget Mold. (c) View of material evaluated over 60% of each tetrahedron's domain. (d) View of material evaluated over 40% of each tetrahedron's domain.

The storage costs for the Widget Mold in the various tetrahedral mesh representations are:

$$
\begin{aligned}
S_{te_{00}} &= 64n_{tetrahedra} + 152 + S_{ms} \text{ bytes} \\
S_{te_{01}} &= 136n_{tetrahedra} + 80 + S_{ms} \text{ bytes} \\
S_{te_{10}} &= 84n_{tetrahedra} + 164 + S_{ms} \text{ bytes} \\
S_{te_{11}} &= 156n_{tetrahedra} + 92 + S_{ms} \text{ bytes}
\end{aligned}
$$

Generalized decomposition



(a) (b)

Figure 8-36: (a) FGMDomain vertices and edges into which the Widget Mold is decomposed in a generalized data structure. (b) A single FGMDomain representing the interior of the Widget Mold in a generalized data structure.

In the previous cases, objects were decomposed into FGMDomains that exactly and explicitly defined the intended material variation ($\mathbf{m}(\mathbf{x})$). For this model, the concept of a procedural FGM-Domain is introduced, in which the composition function is not represented explicitly but must be evaluated relative to the geometry and topology of the model, similar to how offset surfaces can be represented within existing exchange standards [30]. To accomplish this, the function $\mathbf{m}^{proc}(\mathbf{x})$ is

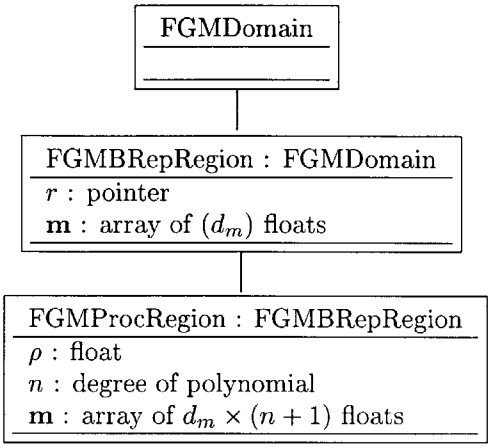


Figure 8-37: Hierarchy from FGMDomain to FGMProcRegion.

associated with a single FGMDomain representing the interior or the model, where

$$\mathbf{m}^{proc}(\mathbf{x}) = \begin{cases} \begin{bmatrix} 0.7 + \frac{3\rho}{50} \\ 0.3 - \frac{3\rho}{50} \\ 0.0 \end{bmatrix} & \text{for } \rho \leq 5 \text{ mm}, \\ \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} & \text{for } \rho > 5 \text{ mm} \end{cases} \tag{8.22}$$

The parameter ρ is the minimum distance of point $\mathbf{x}$ from the object's boundary, as defined by the collection of lower dimensional FGMDomains contained in the relational database maintaining the part's topology. Figure 8-37 illustrates the derivation of the Procedural FGMDomain from the FGMBRepRegion class.

The storage cost for the Procedural FGMDomain is simply the cost of the FGMBRepRegion domain defined in Figure 4-19, plus the storage required to maintain the floating point coefficients and degree of the polynomial as well as the cut-off distance in Equation 8.22; an additional seven floating-point numbers and one integer in this case ($n = 1$, $d_m = 3$).

$$S_{prcr} = S_{int} + [d_m(n+2)+1]\, S_{flt} + S_{ptr} \tag{8.23}$$

The FGMDomains needed to exactly represent this object are listed in Table 8.12. The FGMDomains of dimensions zero and one are shown in Figure 8-36(a), along with the procedural FGMDomain for the model's interior (an FGMProcRegion) in Figure 8-36(b).

The topology for the model may be maintained in either a Cell-Tuple-Graph or Radial Edge data

Classes	*instances*	$\frac{ctgs}{class}$	$\frac{cells}{class}$	$\frac{tuples}{class}$	$\frac{Storage(bytes)}{class}$
FGMPoint	29		29		$174S_{flt}$
FGMRationalBézierCrv					
$n_x = 1, n_m = 0$	46		46		$92S_{int} + 552S_{flt}$
$n_x = 2, n_m = 0$	12		12		$24S_{int} + 192S_{flt}$
$n_x = 3, n_m = 0$	2		2		$4S_{int} + 40S_{flt}$
FGMRationalBézierTri					
$n_x = 1, n_m = 0$	4		4	48	$8S_{int} + 64S_{flt}$
$n_x = 2, n_m = 0$	8		8	96	$16S_{int} + 224S_{flt}$
FGMRationalBézierQuad					
$(m_x, n_x) = (1,1)$, $(m_m, n_m) = (0,0)$	13		13	208	$52S_{int} + 260S_{flt}$
$(m_x, n_x) = (1,2)$, $(m_m, n_m) = (0,0)$	5		5	80	$20S_{int} + 140S_{flt}$
$(m_x, n_x) = (1,3)$, $(m_m, n_m) = (0,0)$	3		3	48	$12S_{int} + 108S_{flt}$
FGMBRepRegion	1	1	1		$3S_{flt} + S_{ptr}$
FGMProcRegion	1		1		$S_{int} + 10S_{flt} + S_{ptr}$
CellTupleGraph		1			$2S_{ptr}$
Cell			124		$248S_{int} + 248S_{ptr} + S_{ms}$
Tuple				480	$480S_{int} + 4320S_{ptr}$
TOTAL					$957S_{int} + 1767S_{flt} + 4572S_{ptr} + S_{ms}$ 36252 bytes + S_{ms}

Table 8.12: FGMDomain and Cell-Tuple-Graph objects required to represent FGM Widget Mold exactly and the associated storage cost.

structure. The storage costs associated with each method are listed in Tables 8.12 and 8.13.

Observations

The storage costs for the approaches to modeling the Widget Mold are graphed in Figure 8-38 and 8-39.

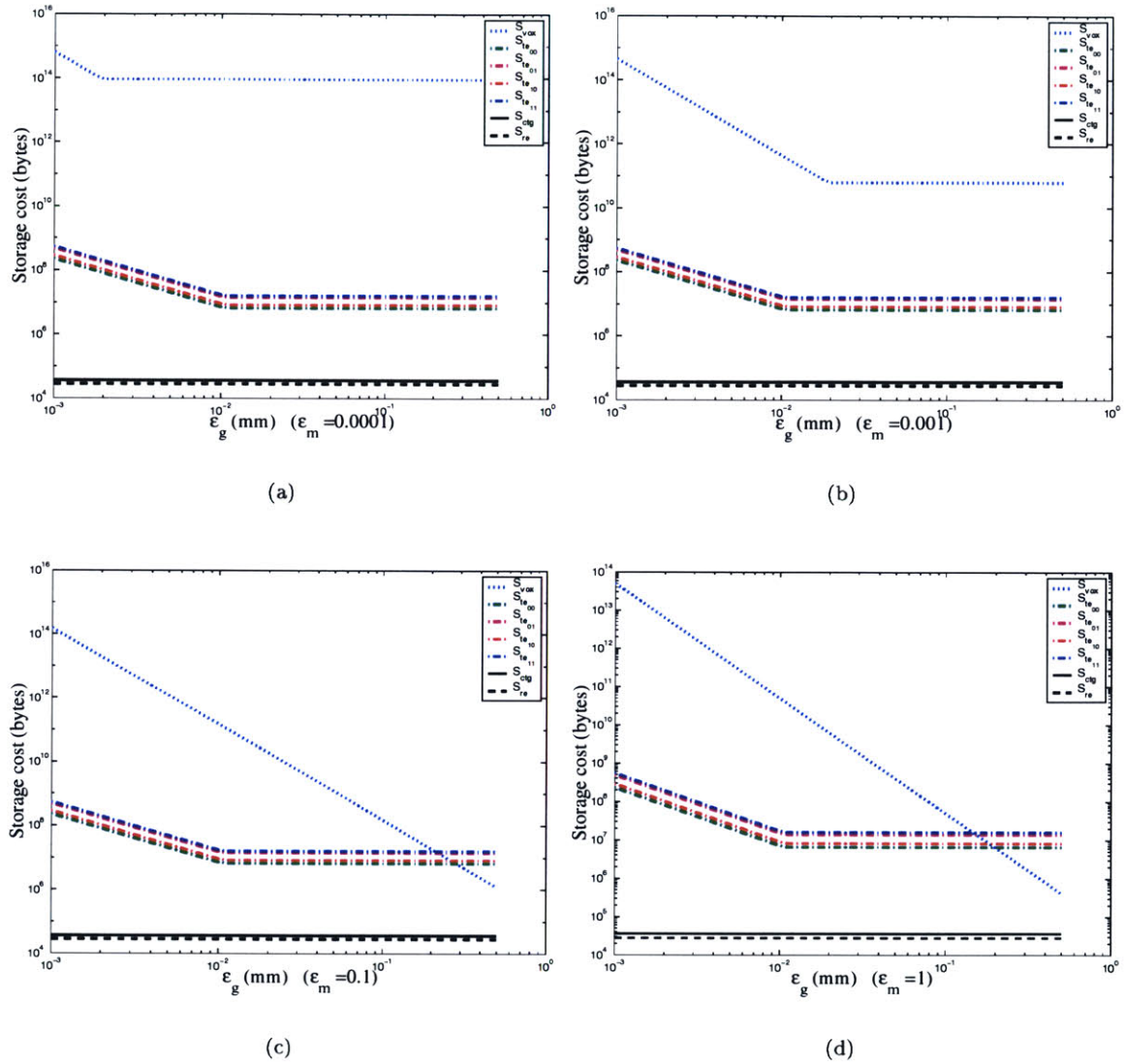


Figure 8-38: Graph of storage cost for representing the Widget Mold (with composition graded from the boundary) as functions of geometric accuracy (a) $\epsilon_m = 10^{-4}$, (b) $\epsilon_m = 10^{-3}$,(c) $\epsilon_m = 10^{-1}$, and (d) $\epsilon_m = 1$.

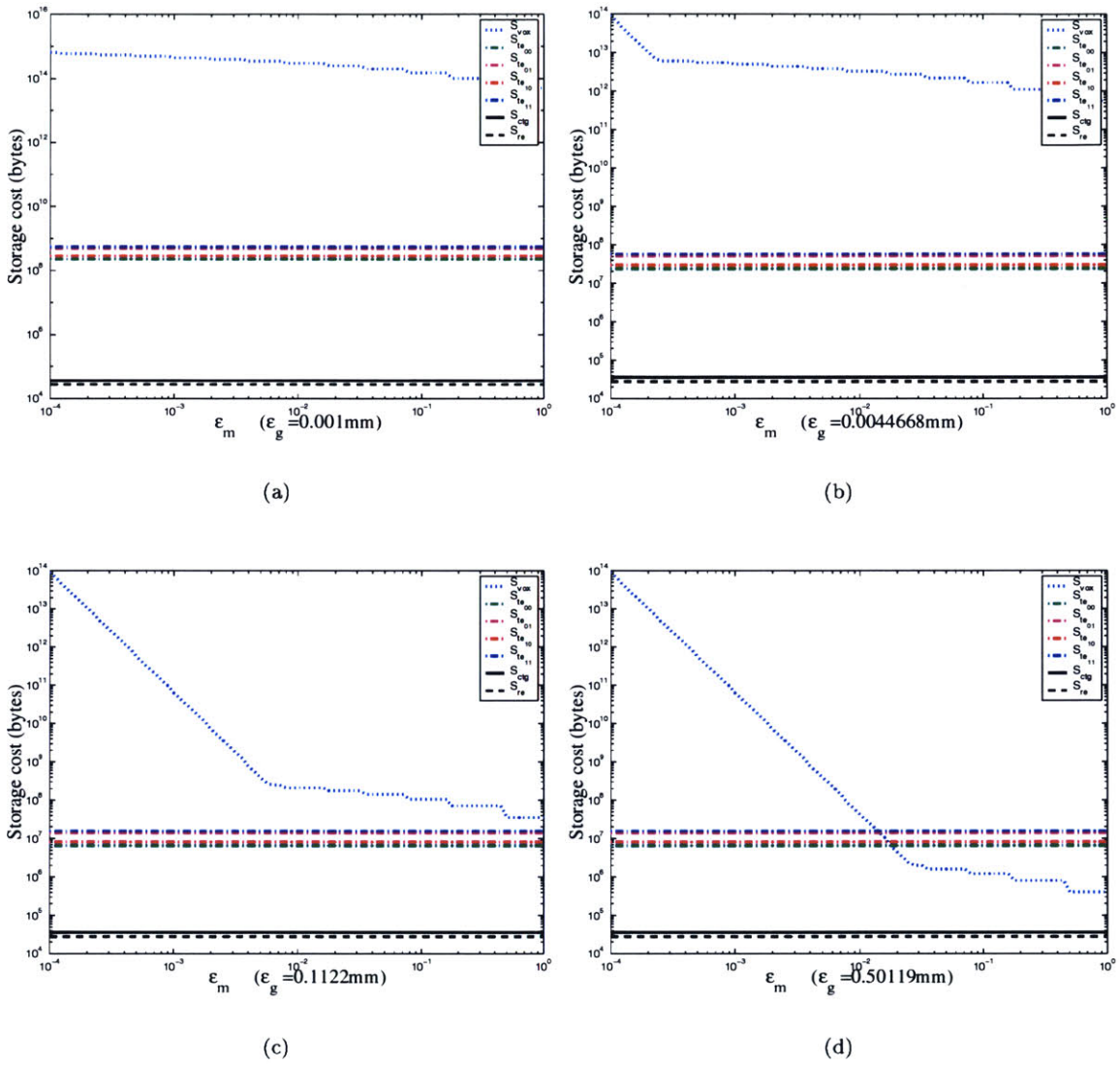


Figure 8-39: Graph of storage cost for representing the Widget Mold (with composition graded from the boundary) as functions of material accuracy (a) $\epsilon_g = 10^{-3}$mm, (b) $\epsilon_g = 0.00446688$mm, (c) $\epsilon_g = 0.1122$mm, and (d) $\epsilon_g = 0.50119$mm

Classes	*instances*	$\frac{Storage(bytes)}{class}$
FGMDomains		$229S_{int} + 1767S_{flt} + 2S_{ptr}$
Complex	1	$S_{ptr} + S_{ms}$
Region	2	$10S_{ptr}$
Shell	2	$8S_{ptr}$
Face	33	$66S_{ptr}$
Loop	33	$33S_{ptr}$
Edge	60	$120S_{ptr}$
Vertex	29	$58S_{ptr}$
FaceUse	66	$396S_{ptr}$
LoopUse	66	$396S_{ptr}$
EdgeUse	192	$1344S_{ptr}$
VertexUse	192	$768S_{ptr}$
TOTAL	$229S_{int} + 1767S_{flt} + 3202S_{ptr} + S_{ms}$	
	27860 bytes + S_{ms}	

Table 8.13: Radial-Edge objects required to represent the topology of FGM Widget Mold exactly and the associated storage cost.

8.3 Discussion

This chapter presented several hypothetical FGM objects and quantified their storage costs in terms of accuracy for the objects' representation in the various data structures outlined in the previous chapters. Although the objects are relatively simple in complexity, they contain features that many real objects might have, including curved surfaces, regions of uniform and graded composition, and features such as holes or internal primitives. They also serve to illustrate how FGM models resulting from the proposed design tools might be represented. In this way, the lessons learned from this analysis apply equally well to *real* models with more of the features presented.

The first object, described, a unit sphere of uniform material composition, introduces how the various data structures represent an object. Although trivial in complexity, its inclusion in this chapter demonstrates how much more efficiently a generalized decomposition modeling method (based on either the Radial-Edge or Cell-Tuple-Graph structures) is able to accurately describe the intended geometry than the voxel-based or triangulated shell approaches.

The subsequent examples demonstrate that the same trend holds true for objects with graded compositions, proving that a generalized modeling method incorporating a suitable library of geometric and material representations (Bézier curves, surfaces, and regions in this case) is the preferred approach to modeling FGM objects in terms of storage cost.

The storage costs for the generalized data structures are constant with the desired accuracy of representation. The storage costs only grow with the number of features present in the model, as shown by the case of the drug delivery device. For each new drug primitive introduced into the model, the storage cost grows by some constant. Therefore, the memory requirements for a model stored in a generalized data structure would be proportional to the number of *features* the designer wishes to include in the model, not the *accuracy* in representation.

The final example, the "Widget Mold", illustrates how a generalized decomposition method could be extended to include additional data classes to further improve its efficiency. In this case, a procedural region class was introduced, efficiently allowing the definition of a material grading from the boundary. To accomplish this, a polynomial function of distance is associated with the region object. Instead of defining the composition explicitly as a parametric mapping into Material Space, the composition at a point is evaluated as a function of its distance to the nearest boundary.

Finally, it is important to note that the storage costs presented here are *bounds* for the memory growth as functions of geometric and material accuracy, assuming uniform meshes. Obviously, adaptive subdivision schemes [11, 21, 22, 63] could be investigated to reduce the memory costs (as well as compression techniques such as octrees [19, 37]), but these issues are beyond the scope of this thesis.

Chapter 9

Conclusions and Recommendations

9.1 Conclusions

New developments in SFF processes promise the capability of fabrication with Local Composition Control, permitting the realization of Functionally Graded Material parts and tools, in which the composition of the objects may consist of spatially varying compositions. This will open the door to the possibility of fabricating a whole new class of parts in which the material variation is chosen to optimize performance and the part is directly fabricated from the CAD model. To realize this potential, however, current CAD/CAM methods need to be extended to enable designers and researchers to model, design, and exchange FGM objects. The first major obstacle that needs to be addressed is the selection of a data structure to hold information about the model's geometry and composition efficiently. Ideally, the chosen data structure will enable the accurate capture of design intent in a form that can be freely exchanged between CAD systems and transmitted to various manufacturing processes.

To motivate this research, limitations in the current practice of processing models through the STL format were presented. Some of the drawbacks include its expense at conveying geometric information, lack of explicit topological information, and the introduction of gaps, holes, and intersections through non-robust processing algorithms. The STEP standard was then presented as a more complete and robust exchange standard, adopted by a wider community for the exchange of product information over a distributed environment. The part of STEP focused on solid modeling was outlined, illustrating the many methods incorporated into the data standard for representing geometry and the connectivity between the topological entities. It was noted, however, that STEP presently makes no allowance for graded material information.

Next a paradigm for information flow for the processing of FGM models was presented, based on the information flow used in image processing of digital photographs and documents for display on

CRTs or hard-copy output. Outlining the steps of how FGM models should be processed establishes two major points. The first is that an FGM object modeled in a computer is nothing more than a vector valued function, providing a mapping from a Build Space into a Material Space.

$$\text{FGM Model} \rightarrow \mathbf{m}(\mathbf{x})$$

Therefore, the ultimate goal of any modeling method is the specification of the function $\mathbf{m}(\mathbf{x})$ by a designer. There are many ways of accomplish this, but each is simply an approach to accurately and efficiently defining $\mathbf{m}(\mathbf{x})$.

The second major point regarding the information flow is the clear separation between the modeling of the FGM object and its processing. The designer should work to define the function $\mathbf{m}_{in}(\mathbf{x})$ within the environment that can most accurately capture his intent. Once this function is defined, it is sampled and converted to a digital representation, analogous to image processing, for fabrication through an SFF process capable of LCC. During this process planning stage, instructions to fabricate the part to the designer's specification are generated. Process parameters, such as the placement of binder droplets in 3D Printing, now enter the information flow. The designer should not be concerned with these parameters, since these are unique to each process. The intended design $\mathbf{m}_{in}(\mathbf{x})$ specifies the desired composition for the fabricated object. In 3D Printing, for instance, the introduction of a binder into the powderbed is usually just a means to bind powder together and is burned out of the finished product during a sintering stage. The shape and composition $\mathbf{m}_{out}(\mathbf{x})$ of the final product (what is left after the binder is burnt out) is the only thing that matters to the designer. Furthermore, it may be possible for the design for an FGM object to be fabricated through multiple SFF systems, each with their own set of process parameters. By requiring $\mathbf{m}_{in}(\mathbf{x})$ to reflect the composition of the final product, not the process parameters, the model may be fabricated through any of the available SFF processing methods. Following this paradigm will enable the vision of a clean separation between between the design and manufacture of FGM objects and encourage the neutral exchange of FGM object models between designers, researchers, and manufacturers, promoting exploration of this new technology.

With the motivation for research in a method to represent FGM design intents, its representation as a vector valued function, and a clean separation between design and fabrication established, several modeling methods are proposed. These range from a voxel-based scheme to a completely general data structure (Radial-Edge or Cell-Tuple-Graph). Each data structure is capable of representing material information by decomposing the model of an object into sub-regions but beyond that, these approaches to modeling FGM objects diverge. The voxel-based approach, for instance, uses a lattice of cubes assumed to be of piece-wise uniform composition, while the tetrahedral modeling approach interpolates field values at nodes in space. With each data structure, the minimal data classes

required to define $\mathbf{m}(\mathbf{x})$ are described and an expression for the storage cost is formed in terms of the number of instances of each class.

The generalized schemes represent the geometric and composition information over cells through FGMDomains. Only a limited subset of FGMDomain classes were introduced here, based on the Bernstein polynomials for providing a parametric mapping into a Build Space and a Material Space. This approach to modeling FGM objects extends the current paradigm for B-Rep solid modeling (in which a topological data structure is used to maintain the connectivity between a collection of face, edge, and vertex geometries bounding a region) to representing cells of graded composition within a generalized data structure. Conceptually, this is similar to Kumar *et al*'s approach to representing FGM objects as collections of r_m sets [40]. It is important to note, however, that the FGM modeling approach presented in this dissertation is based on an adapted Radial-Edge data structure, because of its wide adoption in the IGES and STEP standards. This adapted Radial-Edge data structure is extended to include mathematical descriptions of the material variation over each geometric entity. In addition, the geometric and material descriptions were also associated with the topological entity *Region* of this data structure. The description of the *Region* entity in this way is equivalent to the concept of an *atlas* presented by Kumar *et al.* In order to provide concrete examples in this dissertation, Bézier volume formulations are introduced to define a set of FGMDomains and then used to model several hypothetical models. With both the IGES and STEP standards (as well as many proprietary systems) based on the Radial-Edge data structure, the extension of existing standards to include graded material information should be possible. The Cell-Tuple-Graph data structure was introduced as an abstraction of the Radial-Edge data structure, providing a more concise and simpler structure for maintaining the same information.

With the introduction of data structures for FGM modeling, several design approaches are introduced. These tools are based on concepts from geometric modeling and serve to illustrate how one might define the graded nature of an FGM object. Their implementation, however, depends upon the underlying data structure chosen to represent $\mathbf{m}(\mathbf{x})$.

Since both the voxel-based and mesh-based modeling methods approximate design intent, trends for the sizes of the voxel lattice or mesh were then established in terms of the desired geometric and material accuracy of the representation. These trends were based on the nature of the intended design and include properties such as rate of material variation, surface curvature, material curvature, and minimum feature sizes. For the meshed-based approaches, the expressions given represent upper bounds on the number of elements required to accurately capture the designer's intent since uniform meshes were assumed in the theoretical analysis.

Finally, several hypothetical FGM objects were introduced and their storage costs estimated. These objects, although simple in nature, represent features that would be present in real models, including non-linear surfaces, holes, protrusions, regions of uniform composition, and regions of

spatially graded composition. In each case, the generalized cellular decomposition approach was the most efficient at capturing the intended design. The finite element approach was next best, followed by the voxel-based method. From the point of view of memory requirements, it is recommended that modeling FGM objects as generalized, cellular decompositions be adopted as the paradigm for the representation of FGM models and their design and exchange. By following this approach to FGM modeling, the accepted standard of modeling of solids as B-reps can be extended, a designer's intent is most accurately captured, and the future expansion of a data structure is possible through the introduction of new classes (FGMDomains) for mathematically describing material variation.

9.2 Contributions

Through this dissertation, the following contributions to CAD were made:

1. An information pathway for processing FGM objects based on image processing was introduced. This pathway establishes a clear separation between design of FGM objects, their processing, and their fabrication. Similar to how an image is represented by a continuous vector valued function of the intensity of the primary colors over a two-dimensional space, an FGM object is represented by a vector valued function spanning a Material Space, defined over the three-dimensional Build Space. Therefore, the Model Space for FGM objects consists of a Build Space and a Material Space. The task of modeling and designing an FGM object, therefore, is simply to accurately represent the function $\mathbf{m}(\mathbf{x})$ where $\mathbf{x} \in$ Build Space.

2. Data structures for representing FGM objects were described and analyzed, including a voxel-based structure, a finite element method, and the extension of the Radial-Edge and Cell-Tuple-Graph data structures with FGMDomains in order to represent spatially varying properties. All of the methods are capable of defining the function $\mathbf{m}(\mathbf{x})$ but each does so in a different way. Along with introducing each data structure, the storage cost for each was derived in terms of the number of instances of each of its fundamental classes required to represent an object.

3. In order to determine the optimal data structure to model FGM objects, the storage cost for each of the methods was predicted for several hypothetical models. Although these models were simple in nature, their curved geometries and regions of both piece-wise constant and non-linearly graded compositions reflect the features expected to be found in *real* applications. In each case, the generalized cellular methods were found to be optimal, accurately representing the intended design.

9.3 Future work and recommendations

9.3.1 Investigation into generalized FGM modeler

This thesis indicates that a generalized cellular decomposition approach to modeling FGM objects in terms of FGMDomains is a most memory efficient means to capturing the designer's intent. The work presented in this dissertation, however, assumed a static model with the object already decomposed into FGMDomains. The next step is to establish a data structure based on these concepts. Methods to translate existing solid models into this data structure without loss of accuracy need to be explored, in addition to methods for modifying the data structure to allow the accurate decomposition of models into suitable FGMDomains, accurately capturing design intent. It is important to note that the generalized data structures presented in Section 4.6 closely resemble the B-rep data structures used in many commercial modeling systems. With further investigation in this area, it is hoped that exisiting systems based on B-rep data structures might be extended to include FGMDomains, enabling FGM modeling and design through exisiting CAD systems. With the extension of such systems, tools for design, visualization, interrogation, and processing could then be incorporated as they are developed, helping to realize the potential of FGM design for SFF with LCC.

9.3.2 Exploration of FGMDomains

This dissertation introduced a limited set of FGMDomains in Section 4.6.2 for representing composition variation over curves, surfaces, and regions in terms of Bernstein polynomials. Just as the STEP standard includes a wide range of classes for representing geometry (see Figure 2-4), additional FGMDomains can be defined, including NURBS regions and generalized cylinders or sweeps. The efficient design and interrogation of compositions in terms of these FGMDomains, however, remain open issues. Bézier curves, for instance, have a well known set of properties (such as convex hull and subdivision) that make them appealing for use in design systems. Investigation of similar properties which can be exploited for the design and interrogation of material variations is suggested.

9.3.3 FGM object design methods

Methods to design FGM objects need to be explored. Several design methods (such as FGM fitting, chamfering, filleting, blends, and library) were proposed in this work to motivate the example models for memory analysis. These methods closely parallel methods currently used for geometric design. It is hypothesized that most geometric design operations have analogous operations for FGM design, including lofting, filleting, chamfering, and sweeps. The complete definition and application of these methods, however, remains an open issue.

Investigation into this area is closely related to the underlying data structure chosen to model the FGM objects. The information available for a specific design method and its execution is different for each data structure. However, approaches to applying a proposed design method could be developed for any of the modeling methods, demonstrating the concepts behind each FGM design method.

9.3.4 Establishment of a design methodology for FGM objects

Along with investigating individual design methods, these design methods need to be placed in the context of an overall design process for FGM objects. This involves investigating how a designer might interact with the model and establishing a design methodology. In existing solid modeling systems, for example, objects are created through a sequence of extrusions, cuts, lofts, revolutions, etc. The history of the design session is usually recorded, allowing the designer to "undo" operations and modify the design as desired. Material information in these systems is assigned to regions as attributes, as described in Section 3.2.1.

For FGM modeling, the specification of graded material information is closely related to the geometric design. Consider, for instance, the design of composition as a function of distance from the boundary. If the boundary is modified, the distribution of material within the model should also changed. Ideally, a design system would allow the user to interactively modify the geometry and the composition in any order and with the same ease of use afforded by existing CAD systems for shape definition. How to achieve this goal remains an open issue and depends on the underlying representation as well as the chosen design methods.

A finite element-based modeling system, for instance, requires a traditional B-rep to be decomposed into elements. Once meshed, the relationship between the operations used to define the geometry and the mesh data is unclear. Should geometric operations be performed on the mesh or on the original model? If on the mesh, how are the geometric features represented and how is the mesh modified with each operation? If the geometric operations are performed on the B-rep model, any work done in designing the composition would be lost each time the geometry is modified unless a method to relate the old mesh to the new mesh through the modified B-rep model is established.

One way around this problem would be to extend an existing CAD system based on generalized B-rep or cellular data structure to include the concept of FGMDomains, as introduced in Section 4.6. This should be a modest extension to the underlying data structure in existing commercial CAD systems. In this way, the geometry and the material distribution would both be defined within the same data structure, eliminating the need to correlate data between two different models and design systems. However, the issue of relating the operations in composition design to the data structure must still be addressed. Although it is hypothesized that most geometric operations will have an analogous material operation, the formulation of the steps necessary to realize this potential and their execution in the context of a complete design session (from concept to modeled object) need to

can be understood independently of the representation of the CAD model. An investigation into this area permits an understanding of the material system's *Physical Reconstruction Function* from which an optimal halftoning strategy can be selected. Preliminary work in this direction is reported in [79].

9.3.8 Exploration of halftoning strategies

Along with the investigation of material systems, methods to halftone models must be investigated. These may range from the application of simple dithering lattices to error diffusion algorithms. This area of research is closely related to the previous one (exploration of material systems) since the quality of the printed material is a function of the halftoning strategy and the halftoning strategy should be selected based on the Physical Reconstruction Function. In 3D Printing, for instance, the porous nature of the powder may result in a capillary action for regions of lower density binder printing. This mechanism may distort the distribution of binder in the print bed, resulting in a distribution different than that intended by the designer. The halftone strategy should compensate for this, so that the physically reconstructed object most closely resembles the intended design (see Figure 3-8).

9.3.9 Exploration of Design Rules

Just as in image processing there is a clean separation of the digital representation of continuous-tone image and its output, a similar separation exists for the fabrication of FGM models. This clean separation, however, does not guarantee that any material distribution can be fabricated. Limitations in the process (primitive resolution, accuracy of primitive placement, etc.) and properties of the material system (maximum volume fraction of each material permitted in presence of each other, etc.) will restrict the allowable nature of the function $\mathbf{m}(\mathbf{x})$. For example, in the 3D Printing process, there is a minimum about of binder that needs to be delivered in order to hold the part together to avoid surface imperfections. Similarly, there are upper bounds on the amount of material that can be delivered through the print head within a liquid vehicle. Finally, there is an upper bound on the rate of change of the composition that a particular LCC process can deliver. In order to assist the designer in defining a material distribution that can be successfully fabricated, the limitations need to be understood and captured as a set of Design Rules. These Design Rules may then be applied at various stages of the information flow presented in Figure 3-8. The maximum allowable volume fraction for each material in the material system, for example, may be used to evaluate the material distribution ($\mathbf{m}_{in}(\mathbf{x})$) at the design stage for Design Rule compliance or even used to restrict the assignment of the material distribution at the time of design. Investigation of Physical Reconstruction Function, on the other hand, could lead to rules that should be applied during the process planning stage to help determine the optimal placement of material primitives (halftoning)

to most accurately reproduce the designer's intent ($\mathbf{m}_{in}(\mathbf{x})$) as a fabricated part ($\mathbf{m}_{out}(\mathbf{x})$).

be addressed. It is hypothesized that with the extension of the underlying data structure in existing CAD systems, the design methods explored as suggested in Section 9.3.3 could be incorporated, thereby enabling both the geometric and material design of FGM objects in same manner that geometric design is currently performed.

9.3.5 Tools for fabricating FGM objects

While this dissertation has focused on the issues of modeling FGM objects, tools for handling material information for creating test FGM parts should also be explored. These tools would define the function $\mathbf{m}(\mathbf{x})$ for special applications in order to test halftoning methods, material systems, and applications for FGM parts. One example could be the design of the composition of an FGM object as a function of distance from the boundary. Similar to how the Widget Tool example in Section 8.2.6 introduced a procedural FGMDomain, compositions designed from the boundary can be defined by simply associating a composition function $\mathbf{m}(r)$ with an STL file representing the object, where $\mathbf{m}$ is the composition as a function of minimum distance r to all triangles listed in the STL file. During evaluation (either for visualization or fabrication), the distance of each query point $\mathbf{x}$ from the boundary is computed as needed.

9.3.6 Efficient methods for voxel-based and finite element models

The analysis presented in this dissertation did not take into account any compression using adaptive subdivision techniques applied to the voxel-based or finite element data structures. Such techniques could be applied in a system based on either of these data structures. A voxel-based system, for instance, may be based on an octree representation or run-length-encoded. For the finite element approach, adaptive subdivision methods designed to place smaller elements in regions of greatest composition change and large elements elsewhere need to be developed. Pyramid and brick elements could also be introduced to the system. The efficient use of these structures for design, however, remains an open issue. In addition, access and processing time evaluation of the data structures analyzed here needs to be performed.

9.3.7 Exploration of material systems

This dissertation has addressed the issue of representing the designer's intent efficiently as a solid model. The fabrication of the model, however, remains an open-ended issue. To address this, one of the first tasks is to establish material systems out of which Local Composition Control can be achieved. These might include color-based systems for creating visualization models or varying density systems to create porous models (stainless steel and voids, for instance) to more complicated systems for drug delivery devices. How to locally control the composition in a given material system

Bibliography

[1] L. Bardis and N. M. Patrikalakis. Topological structures for generalized boundary representations. MITSG 94-22, MIT Sea Grant College Program, Cambridge, Massachusetts, September 1994.

[2] K. J. Bathe. *Finite Element Procedures.* Prentice-Hall, 1996.

[3] J. Beaman, J. Barlow, D. Bourell, and R. Crawford. *Solid Freeform Fabrication : a New Direction in Manufacturing: with Research and Applications in Thermal Laser Processing.* Kluwer Academic Publishers, Boston, 1997.

[4] J. H. Bøhn and M. J. Wozny. Automatic CAD-model repair: Shell-closure. In *Proceedings of the Symposium on Solid Freeform Fabrication*, pages 86–94. Department of Mechanical Engineering, University of Texas at Austin, 1992.

[5] J. H. Bøhn and M. J. Wozny. A topology-based approach for shell-closure. In P. R. Wilson, M. J. Wozny, and M. J. Pratt, editors, *Geometric Modeling for Product Realization*, pages 297–318. Elsevier Science Publishers BV, 1993.

[6] D. L. Bourell, R. H. Crawford, H. L. Marcus, J. J. Beaman, and J. W. Barlow. Selective laser sintering of metals. In *Proceedings of the 1994 ASME Winter Annual Meeting*, pages 519–528, Chicago, IL, November 6-11 1994.

[7] E. Brisson. *Representation of d Dimensional Geometric Objects.* PhD thesis, Department of Computer Science and Engineering, University of Washington, Seattle, 1990.

[8] E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete and Computational Geometry*, 9:387–426, 1993.

[9] V. Chandru, S. Manohar, and C. Prakash. Voxel-based modeling for layered manufacturing. *IEEE Computer Graphics and Applications*, 272:42–47, November 1995.

[10] C. Chi. Process insight about LOM systems. In D. L. Bourell et al., editor, *Solid Freeform Fabrication Symposium*, pages 515–522, Austin, TX, August 1996. The University of Texas.

[11] T. Cho, W. Maekawa, N. M. Patrikalakis, and J. Peraire. Topologically reliable approximation of trimmed polynomial surface patches. *Graphical Models and Image Processing*, 61(2):84–109, March 1999.

[12] W. Cho, T. Maekawa, and N. M. Patrikalakis. Topologically reliable approximation of composite Bézier curves. *Computer Aided Geometric Design*, 13(6):497–520, August 1996.

[13] A. Curodeau. *Three Dimensional Printing of Ceramic Molds with Accurate Surface Macro-Textures for Investment Casting of Orthopaedic Implants.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, September 1995.

[14] L. Fang and D. C. Gossard. Multidimensional curve fitting to unorganized data points by nonlinear minimization. *Computer Aided Design*, 27(1):48–58, 1995.

[15] G. Farin. From conics to NURBS: A tutorial and survey. *IEEE Computer Graphics and Applications*, 12(5):78 –86, September 1992.

[16] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, San Diego, CA, 3rd edition, 1993.

[17] J. R. Fessler, R. Merz, A. H. Nickel, and F. B. Prinz. Laser deposition of metals for shape deposition manufacturing. In D. L. Bourell et al., editor, *Solid Freeform Fabrication Symposium*, pages 117–124, Austin, Texas, August 12-14 1996. The University of Texas.

[18] D. Filip, R. Magedson, and R. Markot. Surface algorithms using bounds on derivatives. *Computer Aided Geometric Design*, 3(4):295–311, August 1987.

[19] J. D. Foley, A. Van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, Reading, MA, 2nd edition, 1990.

[20] A. E. Giannakopoulos, S. Suresh, M. Finot, and M. Olsson. Elastoplastic analysis of thermal cycling: Layered materials with compositional gradients. *Acta Metallurgica et Materialia*, 43:1335–1354, April 1995.

[21] H. N. Gursoy and N. M. Patrikalakis. An automated coarse and fine surface mesh generation scheme based on medial axis transform, part I: Algorithms. *Engineering with Computers*, 8(3):121–137, 1992.

[22] H. N. Gursoy and N. M. Patrikalakis. An automated coarse and fine surface mesh generation scheme based on medial axis transform, part II: Implementation. *Engineering with Computers*, 8(4):179–196, 1992.

[23] E. L. Gürsöz, Y. Choi, and F. B. Prinz. Vertex-based representation of non-manifold boundaries. In M. J. Wozny, J. U. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 107–130, Holland, 1990. Elsevier Science Publishers.

[24] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design.* A. K. Peters, Wellesley, MA, 1993. Translated by L. L. Schumaker.

[25] C.-Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling: Part I, Representations. *Computer Aided Design*, 28(10):807–817, October 1996.

[26] C.-Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling: Part II, Boundary evaluation. *Computer Aided Design*, 28(10):819–830, October 1996.

[27] IGES/PDES Organization, U.S. Product Data Association, Fairfax, VA. *Digital Representation for Communication of Product Definition Data, US PRO/IPO-100, Initial Graphics Exchange Specification (IGES) 5.2*, November 1993.

[28] American National Standards Institute. *Product Data Exchange Using STEP (PDES) Part 42, Integrated generic resources: geometric and topological representation.* Fairfax, VA, February 1995.

[29] International Organization for Standardization. *Part 1: Overview and fundamental principles*, 1994.

[30] International Organization for Standardization. *Part 42: Integrated generic resources: Geometric and topological representation*, 1994.

[31] I. Jackson, H. Xiao, M. Ashtiani, and L. Berben. Stereolithography model in presurgical planning of craniofacial surgery. In D. L. Bourell et al, editor, *Solid Freeform Fabrication Symposium*, pages 9–14, Austin, Texas, August 12-14 1996. The University of Texas.

[32] T. R. Jackson, H. Liu, N. M. Patrikalakis, E. M. Sachs, and M. J. Cima. Modeling and designing functionally graded material components for fabrication with local composition control. *Materials and Design*, 20(2/3):63–75, June 1999.

[33] T. R. Jackson, N. M. Patrikalakis, E. M. Sachs, and M. J. Cima. Modeling and designing components with locally controlled composition. In D. L. Bourell et al, editor, *Solid Freeform Fabrication Symposium*, pages 259–266, Austin, Texas, August 10-12 1998. The University of Texas.

[34] P. F. Jacobs. *Rapid Prototyping and Manufacturing : Fundamentals of Stereolithography.* Society of Manufacturing Engineers, Dearborn, MI, 1st edition, 1992.

[35] K. J. Jakubenas, J. M. Sanchez, and H. L. Marcus. Multiple material solid free-form fabrication by selective area laser deposition. *Materials and Design*, 19(1/2):11–18, 1998.

[36] H. Jee. *Computer-Aided Design of Surface Macro-Textures for Three Dimensional Printing.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, April 1996.

[37] A. Kaufman, editor. *Volume Visualization.* IEEE Computer Society, Washington, 1991.

[38] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *Computer*, 26(7):51–64, July 1998.

[39] D. Kochan. *Solid Freeform Manufacturing : Advanced Rapid Prototyping*, volume 19 of *Manufacturing Research and Technology.* Elsevier, New York, 1993.

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

[41] V. Kumar and D. Dutta. An assessment of data formats for layered manufacturing. *Advances in Engineering Software*, 28(3):151–164, April 1995.

[42] V. Kumar and D. Dutta. An approach to modeling multi-material objects. In C. Hoffmann and W. Bronsvort, editors, *Fourth Symposium on Solid Modeling and Applications, Atlanta, Georgia, May 14-16, 1997*, pages 336–353, New York, 1997. ACM SIGGRAPH.

[43] V. Kumar and D. Dutta. An approach to modeling and representation of heterogeneous objects. *Journal of Mechanical Design*, 120:659–667, December 1998.

[44] H. Liu. Algorithms for Design and Interrogation of Functionally Graded Material Solids. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, January 2000. In preparation.

[45] T. Maekawa. An overview of offset curves and surfaces. *Computer Aided Design*, 31:165–173, March 1999.

[46] S. Manorhar. Advances in volume graphics. *Computers and Graphics*, 23(9):73–84, August 1999.

[47] A. Marsan, V. Kumar, D. Dutta, and M. Pratt. An assessment of data requirements and data transfer formats for layered manufacturing. Technical Report NISTIR 6216, U.S. Department of Commerce, Geithersburg, Maryland, 1999.

[48] G. M. Nielson, T. A. Foley, B. Hamann, and D. Lane. Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics and Applications*, 11(3):47–55, May 1991.

[49] J. Owen. *STEP: An Introduction.* Information Geometers, Winchester, UK, 1993.

[50] J. Pegna and A. Safi. CAD modeling of multi-modal structures for free-form fabrication, 1998. Presentation at *Solid Freeform Fabrication Symposium*, Austin, Texas, August 12-14, 1998.

[51] B. Pham. Offset curves and surfaces: a brief survey. *Computer Aided Design*, 24(4):223–229, April 1992.

[52] L. Piegl and W. Tiller. *The NURBS Book*. Springer, New York, 1995.

[53] X. Qian and D. Dutta. Features in layered manufacturing of heterogeneous objects, August 10-12 1998. Presentation at *Solid Freeform Fabrication Symposium*, Austin, Texas, August 12-14, 1998.

[54] J. Rambaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[55] D. F. Rogers and J. A. Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill Inc., 1990. Second Edition.

[56] J. R. Rossignac and M. A. O'Connor. SGC: A dimension-independent model for point sets with internal structures and incomplete boundaries. In M. J. Wozny, J. U. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 145–180, Holland, 1990. Elsevier Science Publishers.

[57] E. Sachs, J. Haggerty, M. Cima, and P. Williams. Three-dimensional printing techniques. U.S. Patent No. 5,204,055, April 20 1993.

[58] E. M. Sachs, N. M. Patrikalakis, D. Boning, M. J. Cima, T. R. Jackson, and R. Resnick. The distributed design and fabrication of metal parts and tooling by 3D Printing. In *Proceedings of the 1998 NSF Design and Manufacturing Grantees Conference, Cintermex Conference Center, Monterrey, Mexico*, pages 35–36. Arlington, VA: NSF, January 1998.

[59] E. M. Sachs, E. Wylonis, S. Allen, M. J. Cima, and H. Guo. Production of injection molding tooling with conformal cooling channels using the three dimensional printing process. *Polymer Engineering Science*, submitted.

[60] D. A. Schenck and P. R. Wilson. *Information Modeling: The EXPRESS Way*. Oxford University Press, Oxford, UK, 1994.

[61] H. P. Seidel and A. H. Vermeulen. Simplex splines support surprisingly strong symmetric structure and subdivision. In P. J. Laurent, A. Le Méhauté, and L. L. Schumaker, editors, *Curves and Surfaces II*, pages 1–13. Boston, AK Peters, 1991.

[62] S. Senger. Visualizing and segmenting large volumetric data sets. *Computer Graphics and Applications*, 19(3):32–37, May/June 1999.

[63] K. Shimada and D. C. Gossard. Computational methods for physically-based FE mesh generation. In *Proceedings of the IFIP TC5/WG5.3 8th International Conference on PROLAMAT '92*, Tokyo, June 22-24, 1992.

[64] B. T. Smith and J. Wellington. *Initial Graphics Exchange Specification (IGES), Version 3.0.* National Bureau of Standards, 1986.

[65] Spatial Technology, Inc. *ACIS Geometric Modeler: Technical Overview.* Spatial Technology, Inc., Boulder, CO, 1996.

[66] D. J. Struik. *Lectures on Classical Differential Geometry.* Addison-Wesley, Cambridge, Mass., 1950.

[67] J. A. Thorpe. *Elementary Topics in Differential Geometry.* Springer-Verlag, New York, 1979.

[68] S. T. Tuohy. A visual tool for demonstrating surface curvature. *Computer Applications in Engineering Education*, 5(1):21–27, 1997.

[69] S. T. Tuohy, J. W. Yoon, and N. M. Patrikalakis. Trivariate parametric B-splines for visualization of ocean data. In *Proceedings of Oceans '95: Challenges of Our Changing Global Environment*, volume 3, pages 1601–1608, San Diego, CA, October 1995. MTS/IEEE Oceanic Oceanic Engineering Society.

[70] R. Ulichney. *Digital Halftoning.* MIT Press, Cambridge, 1987. Based on a Ph.D. thesis at M.I.T. in 1986 entitled 'Digital halftoning and the physical reconstruction function'.

[71] R. A. Ulichney. *Digital Halftoning and the Physical Reconstruction Function.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, January 1986.

[72] K. J. Weiler. Boundary graph operators for non-manifold geometric modeling representations. In M. J. Wozny, H. McLaughlin, and J. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 37–66, Elsevier Science Publishers, Holland, 1986.

[73] K. J. Weiler. The radial edge structure: A topological representation for non-manifold geometric modeling. In M. J. Wozny, H. McLaughlin, and J. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 3–36, Elsevier Science Publishers, Holland, 1986.

[74] L. E. Weiss, R. Merz, F. B. Prinz, G. Neplotnik, P. Padmanabhan, L. Schultz, and K. Ramaswami. Shape deposition manufacturing of heterogeneous structures. *SME Journal of Manufacturing Systems*, 16(4):239–248, 1997.

[75] P. R. Wilson. Processing tools for EXPRESS. Technical Report 92003, Rensselaer Design Research Center, Troy, NY, February 1992.

[76] F. E. Wolter and S. T. Tuohy. Approximation of high degree and procedural curves. *Engineering with Computers*, 8(2):61–80, 1992.

[77] S. Wong and K. Hoo. Digital teaching in diagnostic imaging. *Computer Graphics and Applications*, pages 56–65, May/June 1999.

[78] B. M. Wu, S. W. Borland, R. A. Giordano, L. G. Cima, E. M. Sachs, and M. J. Cima. Solid free-form fabrication of drug delivery devices. *Journal of Controlled Release*, 40(1/2):77–87, 1996.

[79] H. Wu, E. M. Sachs, N. M. Patrikalakis, D. Brancazio, J. Serdy, T. R. Jackson, W. Cho, H. Liu, M. Cima, and R. Resnick. Distributed design and fabrication of parts with local composition control. In *2000 NSF Design and Manufacturing Grantees Conference*, Vancouver, BC, Canada, January 2000. http://deslab.mit.edu/3dp/3dppapers.html, http://www.engr.washington.edu/ uw-epp/nsf/.

[80] J. Yoo, K. Cho, W. S. Bae, M. J. Cima, and S. Suresh. Transformation-thoughened ceramic multilayers with compositional gradients. *Journal of the American Ceramic Society*, 81(1):21–32, January 1998.