Feature-Based Design of Solids with Local Composition Control

by

Hongye Liu

B.E. in Precision Machinery and Instrumentation, University of Science and Technology of China, 1993
S.M. in Naval Architecture and Marine Engineering and
S.M. in Mechanical Engineering, Massachusetts Institute of Technology, 2000

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 2004

Author .....
Department of Ocean Engineering
April 8, 2004

Certified by .....
Nicholas M. Patrikalakis
Thesis Co-Supervisor, Kawasaki Professor of Engineering,
Professor of Ocean and Mechanical Engineering

Certified by .....
Emanuel M. Sachs
Thesis Co-Supervisor, Fred Fort Flowers and Daniel Fort Flowers Professor
of Mechanical Engineering

Accepted by .....
Michael S. Triantafyllou, Professor of Ocean Engineering
Chairman, Committee for Graduate Students

Feature-Based Design of Solids with Local Composition Control

by

Hongye Liu

Submitted to the Department of Ocean Engineering
on April 8, 2004, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis presents a parametric and feature-based methodology for the design of solids with local composition control (LCC). A suite of composition design features are conceptualized and implemented. The designer can use them singly or in combination, to specify the composition of complex components. Each material composition design feature relates directly to the geometry of the design, often relying on user interaction to specify critical aspects of the geometry. This approach allows the designer to simultaneously edit geometry and composition by varying parameters until a satisfactory result is attained. The identified LCC features are those based on volume, transition, pattern, and (user-defined) surface features. The material composition functions include functions parametrized with respect to distance or distances to user-defined geometric features; and functions that use Laplace's equation to blend smoothly various boundary conditions including values and gradients of the material composition on the boundaries. The Euclidean digital distance transform and the boundary element method are adapted to the efficient computation of composition functions. Theoretical and experimental complexity, accuracy and convergence analyses are presented. The developed model is a multi-level and graph-based representation, thereby allowing for controls on the model validity and efficiency in model management. The representations underlying the composition design features are analytic in nature and therefore concise. Evaluation for visualization and fabrication is performed only at the resolutions required for these purposes, thereby reducing the computational burden.

Thesis Co-Supervisor: Nicholas M. Patrikalakis, Ph.D.
Title: Kawasaki Professor of Engineering,
Professor of Ocean and Mechanical Engineering
Department of Ocean Engineering and Department of Mechanical Engineering

Thesis Co-Supervisor: Emanuel M. Sachs, Ph.D.
Title: Fred Fort Flowers and Daniel Fort Flowers Professor
of Mechanical Engineering
Department of Mechanical Engineering

Dedication

In memory of my father Guohua Liu.

Acknowledgments

Thank God for His grace and peace and praise Him!

I thank my advisors Professor N. M. Patrikalakis and Professor E. M. Sachs with profound gratitude for their expert guidance, patience and strong support during my thesis work. Not only I learned a lot from them in terms of knowledge and methods in scientific research, but also their passion for truth and creative ideas and decent research ethics gave me great examples of the kind of scholarship I would definitely want to emulate in my career. I also thank the other members of the thesis committee: Professor D. C. Gossard, Professor T. Maekawa and Dr. W. Cho. I am grateful to Professor D. C. Gossard for giving me valuable advice in the formulation of my thesis work and in producing the final thesis. My special thanks go to Professor T. Maekawa who mentored me in various ways, especially by setting a good example for me in our cooperation in the SFF project. I would also like to give thanks to Dr. W. Cho for providing useful extensive comments on my thesis.

In addition, I thank Professor X. Ye who helped me a lot in understanding the SolidWorks system and answered my questions regarding the graph representation. I also thank Dr. T. R. Jackson, who put the first footprint on the FGM field that I followed. He gave me much help in the early phase of my thesis work. My thanks also go to Mr. C. C. Stratton and other members of 3DPrinting Laboratory for their cooperation and help in this thesis project. I am also grateful to Dr. Yuming Liu for providing a computer code for the BEM method and Dr. V. Frayssé et al. for the GMRES codes.

I also cherish the experience of working with all the past or present members of the Design Laboratory: Dr. G. Shen, Dr. G. Yu and Dr. G. V. Papaioannou; Mr. F. Baker, Design Laboratory manager for all the help with technical computer problems; Dr. C. Evangelinos for his expert help with operating systems and all the rest of the Design Laboratory fellows whose friendship made my life at MIT easier.

Financial support for this project was provided in part by the Office of Naval Research (grant #N00014-01-1-1065) and the National Science Foundation (grant #DMI-0100194). CAD models for this thesis were generated with SolidWorks^{TM}.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Solid Freeform Fabrication (SFF) technology, also called Rapid Prototyping is a modern Computer Aided Manufacturing technology through which prototypes, parts, and tools are built in an additive fashion directly from CAD models. Among various SFF processes, the Three-Dimensional Printing (3D Printing) at MIT [66], the Selective Laser Sintering at University of Texas [2] and the Shape Deposition Manufacturing at Carnegie Mellon and Stanford Universities [52] are among the most prominent. 3D Printing [66] is one of the SFF manufacturing processes in which a 3D structure is built layer by layer and completed near-pointwise. Compared with the other SFF manufacturing processes, 3D Printing not only possesses the advantage of producing new complex solids that traditional technologies such as subtractive machining, forming or casting can not make or make efficiently, but also has better flexibility in exercising control over composition.

One of the great potential benefits offered by *Solid Freeform Fabrication* (SFF) technology is the ability to create parts that have composition variation within them. Such *Local Composition Control* (LCC) has the potential to create new classes of components. In this thesis, the acronym "LCC" will be also used to denote "Locally Controlled Composition," depending on the context. Material composition can be tailored within a component to achieve local control of properties (e.g., index of refraction, electrical conductivity, formability, magnetic properties, corrosion resistance, hardness vs. toughness, etc.). By such local

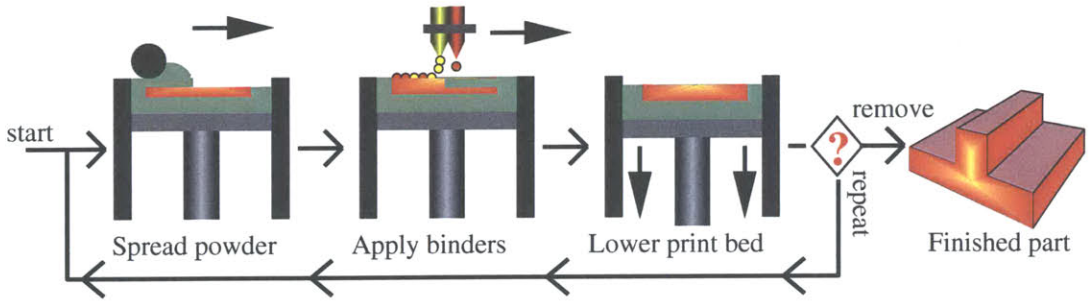


Figure 1-1: 3D Printing illustrating Local Composition Control (LCC)

control, monolithic components can be created which integrate the function of multiple discrete components, saving part count, space and weight and enabling concepts that would be otherwise impractical. Controlling the spatial distribution of properties via composition will allow for control of the state of the entire component (e.g., the state of residual stress in a component). Integrated sensors and actuators can be envisioned which are enabled by LCC (e.g., bimetallic structures, in-situ thermocouples, etc.). Devices which have as their function the control of chemical reactions are possible. The utility of "mesoscopic" parts made by SFF will depend strongly on the ability to locally control composition.

Among the SFF processes, 3D Printing is particularly well suited to the fabrication of parts with LCC. 3D Printing creates parts in layers by spreading powder, and then ink-jet printing materials into the powderbed [64, 66, 67]. In some cases, these materials are temporary or fugitive "glues", but in many cases, these materials remain in the final component. Examples of the latter include: ceramic particles in colloidal or slurry form, metallic particles in slurry form, dissolved salts which are reduced to metal in the powderbed, polymers in colloidal or dissolved form, and drugs in colloidal or dissolved form. 3D Printing has been extended to the fabrication of LCC components by printing different materials in different locations, each through its own ink-jet nozzle(s). Figure 1-1 illustrates this conceptually with three different colors, each representing the printing of a different material into the powder bed with local control of position. 3D Printing is thus capable of fully three-dimensional control of composition.

1.2 Motivation

Several promising applications of 3D Printing are under active development. Drug delivery devices are being created by printing different drugs at prescribed locations within the interior of a pill or implantable device. These drugs are then released into the body according to designed release profiles [38]. Gradient Index Lenses (GRIN) are another class of LCC applications which refract light by gradients in the index of refraction, rather than by external geometry. Such lenses can provide the functionality normally associated with multi-component ground optics at lower cost and in a smaller space. The drug delivery and GRIN applications are for high value added devices which are small in size and thus can reasonably be manufactured by 3D Printing. LCC is also being applied to the fabrication of tooling by 3D Printing. Hard phases such as TiC are being printed local to the surface of a tool for increased wear resistance. Tools with local control of porosity (for venting of gases) are being fabricated by printing a material which acts to block the infiltrant during furnace densification. Although large in size, tooling applications can be economical because small quantities are required.

While one may not hope to match the impact of VLSI fabrication methods on engineering and society, the parallels are intriguing. VLSI and SFF are layer processes. VLSI depends on local control of composition and SFF is capable of the same. Perhaps, as in the case of VLSI, it will be found that designers, given the proper tools, will find uses not now imagined for LCC in SFF.

Realizing the potential utility of LCC in SFF is a challenge because of the limited knowledge, methods and tools in the area of computer representation and design of parts with LCC. Generic computer representations are necessary to allow for electronic specification of composition within a component and it is desirable to devise a suite of tools which allows a designer to communicate with this representation using high level features that are sensible to a designer. The designer must be able to visualize and interrogate the evolving model. The modeler must not allow the designer to request that which cannot be made. Process specific tools include methods to render desired continuous composition profiles in the discretized form required by a specific process, etc. The resulting tools should be generic and

applicable to a broad range of SFF technologies. In this thesis however and in cases where the outcome is process specific, **3D Printing** is used as the prototypical SFF technology.

The CAD modeling system not only needs to provide design tools, but also needs to provide functionality for the user to evaluate the properties of a model. The evaluation of the composition of a model with functionally graded materials (FGM) [34] will be most important for either the visualization or the post-processing of the FGM model for fabrication. Query of the composition may be in the form of the query for a point, a ray, a plane or a grid. Efficient evaluation of the composition functions is very important given that the amount of queries is large and the related geometry can be very complex.

1.3 Thesis organization

Chapter 2 begins with a review of the work that has been done at MIT in the field of modeling, design and the fabrication of components with local composition control. In addition, recent MIT work on LCC modeling and design methods in the published literature is reviewed and the limitations of all existing methods are summarized. Chapter 2 ends with a review of feature-based design and modeling methods in existing literature.

The objective of this thesis is presented in Chapter 3. In addition, a summary of the approaches used is presented.

Chapter 4 contains the definition and classes of local composition control features.

In Chapter 5, the design of material composition functions is presented. For the algorithms for each design method, the complexity and/or convergence is analyzed and comparison with numerial experiments illustrating various versions of the algorithms are given.

Chapter 6 provides the method for maintaining the feature-based LCC model. In addition, algorithms in model creation, editing and validation are presented.

In Chapter 7, efficient evaluation of the composition of a LCC object represented with a LCC feature model is presented. Analysis for the algorithms involved is also described.

Description of the implementation of all the algorithms of Chapter 3 to 7 is given in Chapter 8 with numerical results on several sample models.

Chapter 9 concludes the thesis and provides potential directions for related future work.

Appendix A provides pseudo-code for some of the algorithms developed in this work.

Chapter 2

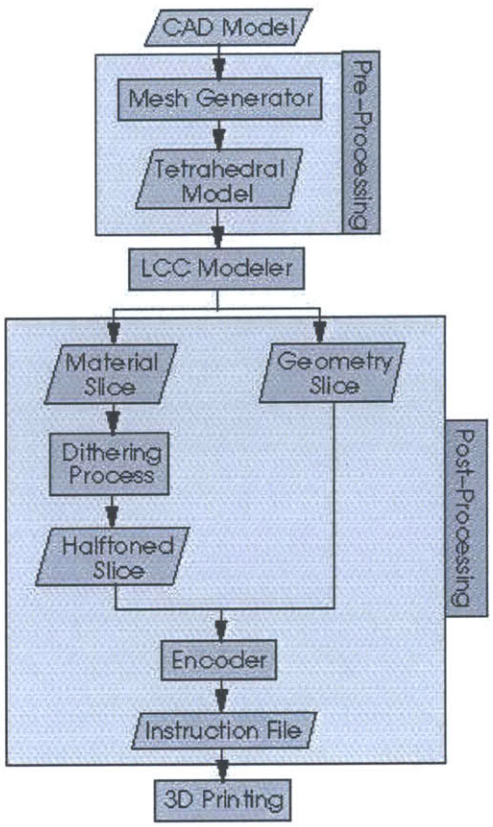
Review

2.1 Past work on design of LCC object

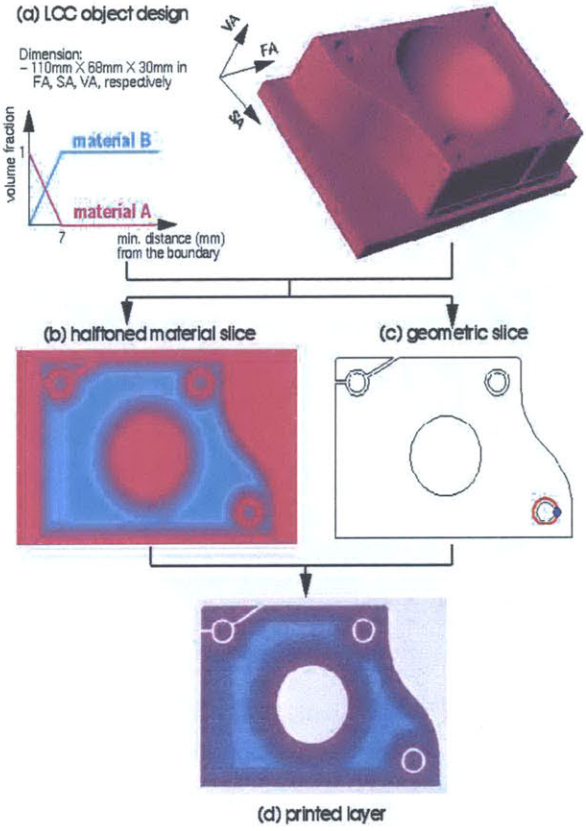
2.1.1 Past MIT work on local composition control

(a) Overview: The work on local composition control at MIT has produced an information pathway for 3D Printing that begins with a designer interacting with a standard CAD system to define the shape of the object, see Figure 2-1-(A). The solid model thus created is then exported from the CAD system in a standard exchange format such as STEP [3] or IGES [32]. An LCC modeler was implemented based on a tetrahedral mesh data structure. This finite-element based LCC modeler can be thought of as a special instance of a generalized cellular decomposition approach to LCC modeling [34, 35, 65]. It was chosen as a convenient method to demonstrate the information pathway and to explore the issues associated with LCC. Once the geometry of the model is fully defined, it is loaded into a finite-element mesh generator via a neutral format, and meshed into a set of tetrahedra. This process is referred to as *pre-processing* in Figure 2-1-(A). The composition of a part is established by specifying the composition values at the vertices of each tetrahedron and interpolating between them. As an *exemplar* of a design tool, a method was developed to specify a composition profile normal to the surface and apply this profile to an entire object [42, 43].

Post-processing then converts the designed LCC model into instructions [78] for the 3D



(A) Information pathway



(B) Demonstration of information pathway

Figure 2-1: Information flow for LCC with 3D Printing

Printing machine. Post-processing takes place on a layer-by-layer basis along two parallel paths: (1) the accurate definition of the surface (Geometry Slice); and (2) rendering the composition of the body (Material Slice). The continuous-tone material composition is rendered into printable discrete elements using halftoning (or dithering) algorithms [19]. The boundary and composition information is recombined to produce the drop-by-drop instructions that are loaded onto the 3D Printing machine. Special attention is given to reconciling conflicts which occur at the boundary where the designer's intent in both composition and surface finish must be recognized [78].

The complete information and 3D Printing pathway was tested and demonstrated [18] with a part of representative complexity as shown in Figure 2-1-(B). The part is an injection molding tool and the design challenge is to place hard phases in a designed composition profile near the surface. As a demonstration, two colors of ink were printed (magenta and

cyan) with the condition that the sum of the material volume fractions was everywhere constant. The bottom image in Figure 2-1-(B) shows a photograph of a layer of the actual printed part. This can be compared with the material and geometry information above it, which become merged to produce the instructions which led to the printed part.

(b) Alternative representations and evaluation: In modeling LCC parts efficiently, any such method should provide a concise and accurate description of all of the relevant information about the part with an affordable cost in terms of storage. Jackson [33] presented and analyzed various representations for modeling objects with LCC in this respect. Since both the voxel-based modeling and finite-element mesh approach 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. The storage costs of the generalized B-rep data structures are constant with the desired accuracy of representation, and grow with the number of features in the model.

(c) Design methods and visualization: It is assumed that the input geometry is a single solid represented via a boundary representation including tessellated models and curved models, obtained from a CAD system, and exchanged via a standard file format such as STL [1], IGES [32] or STEP [3]. The first algorithm developed allows specification of the locally controlled composition as a piecewise polynomial or rational function of the minimum distance d from the entire boundary surface. In order to design the composition as a function of d, an efficient distance transform (DT) is necessary. Among the approaches for an efficient DT, space division via a rectangular lattice is particularly useful and easy to implement. Specifically, the approach for improving efficiency of DT includes pre-processing the model with bucket sorting [21] and digital distance transform of the buckets [42, 43]. Complexity analysis of the algorithm outlined above and experimental results demonstrated effective performance of the method.

Once the designer specifies the composition functions for each material (in terms of the minimum distance and also the footpoint), then the problem reduces to developing algorithms for efficient evaluation of composition at either arbitrary points within the solid or a

sequence of points that exhibit spatial coherence. These methods have possible applications in new turbine blade and heat exchanger designs. With the composition function evaluated effectively, the visualization of the composition is done through various computer graphics techniques. The methods implemented include color-coded point sets, color-coded planar sections, cuberilles, and ray casting of the composition [43, 33].

2.1.2 Literature review on LCC modeling methods

Jackson *et al.* [36] presented a review of representation methods for heterogeneous solids such as voxel- or mesh-based structure, volumetric texture-based structure, and generalized modeling methods. Pratt *et al.* [59] classified the existing approaches into exact boundary-based parameterizations of object interiors, volume discretization approaches and non-boundary conforming parameterization methods. Chandru *et al.* [15] suggested using a voxel-based representation to build composite structures by associating material information with each voxel. Pegna and Safi [58] suggested using a finite-element mesh to represent the model and assign material values to each node of the mesh. Liu *et al.* [42] developed a finite-element based representation system and a distance function-based design method and a related efficient evaluation method. The algorithm developed allows specification of the locally controlled composition as a piecewise polynomial or rational function of the minimum distance d from the entire boundary surface.

Kumar and Dutta [41] presented r_m set-based representation method and a Boolean operation-based material composition function design method. Shin and Dutta [72] extended the work in [41] to a constructive representation scheme. Under such a scheme, r_m sets are not disjoint interiors of the heterogeneous object any longer. In the regions where different sets overlap, the material function is the distance-based weighted sum of the material function of each set. They also proposed a material design function related to the coordinates, and geometric specific blendings of materials and its sweeping [72]. Park *et al.* [57] developed a volumetric multi-texturing method based on a procedural algorithm for evaluating material variation within a model. Their method attaches material blending functions to entities in an existing solid modeling system. Martin and Cohen [51] presented a framework for representing attribute data independently of geometric data within a trivari-

ate NURBS volume. They extended an existing modeling and data fitting techniques and developed efficient algorithms for attribute function evaluation and visualization. Siu and Tan [73] presented a scheme of including a grading source in the design of material composition variation and constructed a design method with the extended CSG-type Boolean operations. Biswas *et al.* [7] presented a heterogeneous material modeling method for solids based on distance fields.

2.1.3 Design of LCC solids

The design of LCC solids is another important issue in an LCC modeling process. In one of the definitions of "Solid Modeler", a solid modeling system is defined as a computer program that provides facilities for storing and manipulating data structures that represent the geometry of individual objects or assemblies [50].

Currently, there are two different categories of LCC design approaches. One is design in top-down fashion, in which the CAD model is decomposed into simpler geometry sub-domains, and then the designer designs graded composition over all the sub-domains. For example, the system developed at MIT [33] provides composition functions, especially a graded composition in terms of volume fractions of the material over the domain of each sub-region. Over each cell's domain c_k, the shape and composition is formulated in terms of a set of control points and control compositions which are blended with the barycentric Bernstein polynomials [30]. The number of control points and the degrees of the control composition blending function are determined according to the degree of variation of the geometry and composition in each cell. With each control composition of the model representing a degree of freedom (DOF), the design of LCC parts reduces to the procedure of assigning values to each DOF and blending over the whole domain. A design tool that helps in assigning control compositions in terms of distance functions to a selected feature is developed [33]. The selected feature may be a fixed reference in the model space, such as a point, line or plane, or a feature of the model, such as a particular face or its entire boundary, or an independent boundary shell in .STL format. After a feature is selected, the designer specifies a variation for the LCC in terms of distance from the feature: $\vec{m}(\vec{x}^*) = \vec{m}(r(\vec{x}^*))$, where r is the distance of a query point x^* from the reference feature. Next, the design

tool automatically visits and assigns the control compositions for each cell, and in this way defines the composition over the whole model domain. The other approach is to design LCC objects via a composition based on a library of predefined components. The composition of different components can be done by using CSG-type Boolean operators specific to the chosen data structures. Examples of this type of approaches are the work by Shin *et al.* [72] and Siu *et al.* [73]. In these two papers, similar methods were presented as to how to use Boolean operator in creating compositions when the predefined components overlap geometrically. Shin *et al.* [72] have based their approach on a CSG tree representation while Siu *et al.* [73] chose an image-based representation. In terms of the design of composition for the primitive components, both papers use reference geometries (fixed reference geometric entities) for distance-based design profiles which are analogous to the method of Jackson *et al.* [34].

2.1.4 Summary of limitations of existing approaches

Current approaches either based on volume meshing or cellular decompositions are awkward in editing geometric and material composition information simultaneously, because they lack the concept of editable LCC features; in effect, they permit sequential editing (first of geometry and then composition), which is not flexible and limits the designer's options. Current LCC models are limited to low level data and operators and do not allow for the symbolic representation of the designer's intent with respect to composition. Also as such, design changes cannot be efficiently propagated. Tessellation of the volume of a model (e.g., via tetrahedral meshing) early in the design and fabrication pathway, although expedient for testing of ideas, does not provide a long-term solution for the following reasons:

1. Tessellation implies both approximation of surface geometry and material composition, which is undesirable in general, and for realistic accuracies of approximation leads to verbose evaluated representations, that are unattractive for general LCC modelers.

2. Tessellation approximation accuracy for surface geometry and material composition can be improved via adaptive meshing procedures, however these are difficult to implement robustly and efficiently.

3. Methods for tessellation of a volume into tetrahedral meshes suffer from the general robustness problem in computational geometry relating to inexact computation.

2.1.5 Review of feature-based design and modeling methods

Feature technology has emerged in response to vital industry needs in design and manufacturing. A recent definition of a *feature* is:

A representation of shape aspects of a product that are mappable to a generic shape and functionally significant for some product life-cycle phase [6].

Compared to the approach of feature recognition, feature-based design is rapid and easier by making use of the information in the process. In addition, from the design point of view, feature-based design has the potential of supporting the design process better, such as improving the quality of design and improving the link between design and applications [69, 70]. With rapid development of feature technology, feature-based design is becoming one of the fundamental design paradigms of CAD systems.

In feature-based design, parts are constructed from a sequence of feature adding operations. This paradigm lends itself to separating the design into two layers, one comprising an unevaluated, generic representation, the other comprising an evaluated, instance representation. Pratt [60] first suggested an explicit volumetric representation of features via extension of the radial-edge data structure [76, 77], a non-manifold boundary representation data structure. Rossignac [62] proposed a cellular scheme that permits mixed-dimension representation with the Selective Geometric Complex (SGC) structure. He presented methods based on space decomposition and the concept of intentional feature to correct validity errors caused by feature interactions. He also addressed the issue of editing form features. Bidarra *et al.* [4] presented a cellular model as an alternative to the SGC model to avoid excessive generality. Cellular model is a connected set of volumetric quasi-disjoint cells. This method models features with cells in the cellular model. Each feature has an explicit volumetric representation, a set of associated cells. Feature interactions are maintained in attributes of cells, cell faces and cell edges. With this cellular model, Bidarra *et al.* have classified feature interactions and developed an algorithm for detecting these interactions. A summary of their work on feature modeling is presented in [6].

In terms of feature definition, the procedural method is very general and convenient specifically for object-oriented programming. In this manner, CAD/CAM integration can be carried out more easily. An example of a neutral procedural definition language is *Erep* by Hoffmann and Joan-Arinyo [27]. Dedhia *et al.* [23] presented the ASU testbed for rapid prototyping of feature-based applications. Each feature has a feature type identifier, a name, a list of generic, compatible features and a CSG tree representation. Hoffmann and Joan-Arinyo [28] presented a procedural mechanism for generating and deploying user-defined-features (UDF) in a feature-based design paradigm. The proposed paradigm is to address customization needs in a simple, effective way. The usefulness of that mechanism relies on three basic capabilities: use of standard tools, parametrization of UDFs, and graphical interaction. The advantage of UDF also lies in that design changes can be prestructured and then compound features can be made to have greater independence from each other so that validity errors induced by feature interaction is less likely to happen. In the area of designing and editing features within the constraint-based framework, Chen and Hoffmann [16] presented semantics for feature attachment; Capoyleas *et al.* [13] developed a schema for generic naming of geometric entities. Based on such a schema, Chen and Hoffmann [17] presented algorithms for naming matching, which is the mapping from abstract features to the corresponding geometric entities in the boundary representation for the purpose of design reevaluation.

By feature-based design, functionality can be captured using constraints as mathematical equations of the variables that the design depends on. An approach for capturing engineering meaning has been described by Nielsen *et al.* [53]. Ullman [75] explored the evolution of function and behavior during the mechanical design process.

The feature-based approach has also been introduced into assembly design. Shah and Rogers [71] presented an assembly model as an extension of feature-based design. They showed several basic structures that can be used to define relationships between assemblies, parts, features, feature volume primitives and evaluated boundaries. Generic relations which facilitate constraint specification between target and reference entities were also presented. Cugini [22] studied the concept of assembly feature and developed a prototype system for application in the aeronautics field. Recently, Brunetti and Golob [12] presented an

approach for conceptual design via extension of a recent feature-based parametric part and assembly modeling system. The functions considered are such that they are mappable to a working principle of assemblies. Holland and Bronsvoort [29] introduced an integrated object-oriented product model for both single-part objects and assemblies of objects and showed the usefulness of such modeling in robotic planning for manufacturing.

Chapter 3

Objective and Summary of Approaches

3.1 Objective

Section 2.1.4 summarizes the limitations of current approaches. In comparison with solid modeling, feature-based modeling maintains high level data in the model and relations among them. The high level entities and their associativity in a feature model provide the user information with engineering significance. Feature-based parametric design provides a promising basis for modeling of parts with gradient material composition. Although the current Feature-Based Design (FBD) systems carry rich information in terms of features, they only allow users to create multi-material solids with piecewise constant composition using composite structures and assemblies.

With an overall goal of reducing the limitations in current LCC object design methods and provide automated methods and tools for LCC design and interrogation, this thesis has the following objectives:

- identification and formalization of features for local composition control (**LCC features**).
- development of LCC feature creation and editing functionalities.
- an extended representation of LCC objects based on an existing feature-based part

and assembly modeler.

- efficient and robust evaluation of LCC objects at different levels of resolution for both visualization and fabrication.

3.2 Summary of approaches

In order to address the limitations summarized in Section 2.1.4, our approach involves the following key ideas:

- By introducing the concept of editable LCC features, the simultaneous editing of geometric and material information is formalized and simplified.

- Maintenance of a procedural unevaluated exact representation for the geometry and composition *for as long as possible* along the information pathway, provides a high level codification of the design useful in data exchange, for the integration of design with downstream applications and in a general setting not associated with a specific SFF process.

- Supply users with controls on validity of a LCC solid and the efficiency in updating the design and evaluation of the design by using a dependency graph-based management model.

- Evaluation of the above exact representation is performed as needed at later stages of the pathway, e.g., for visualization and design verification at an appropriate resolution corresponding to the visualization parameters or for fabrication only at the resolution printable by a particular process.

For the purpose of allowing the user to specify composition variation in the interior of a solid, this thesis defines a LCC feature as a construct that has primarily two attributes: the generic geometric shape and the composition profile defined over it. The reason for combining both shape and composition attributes is for conceptual simplicity of specification of such features from the user's point of view at the time of design – relying on a natural vocabulary of design terms. In terms of data structures, a LCC feature is composed of two substructures, one providing the representation of the generic geometric shape, and the other providing the representation of the composition profile. Therefore, the LCC feature can be viewed as primarily comprising two sub-features, respectively for geometric shape and

composition profile. The geometric sub-feature can be any standard geometric feature or an extension of it by the general user-defined feature (UDF) method [28]. Using feature-based parametric design methods, we develop tools for LCC composition profile design and editing of LCC features by extending a current feature-based design system (SolidWorks [74]). Our approach involves developing a set of composition parametrization methods according to different types of geometric features. A graphical user interface (GUI) is also developed for interactive analytical function assignment with respect to the parameters identified.

SolidWorks [74] is a successful feature-based parametric part and assembly modeling system. Extension based on such a model is developed to allow LCC objects to be represented as a feature-based assembly of quasi-disjoint LCC features. LCC features are an extension of solid compound features (components) as described in previous paragraphs. The extension of the feature model is able to accommodate the representation of composition profile sub-feature and its relation with geometric features at both the component and the assembly levels. Extended feature management graph is also developed for the purpose of controlling the validity, management of the design, editing, processing of the model features at different levels.

Evaluation of LCC objects for visualization and SFF fabrication is an important part of this work. Considering different types of rendering methods and different required resolutions, appropriate intermediate models are constructed. For example, issues related to 3D Printing process are taken into account to accomodate the downstream processing. Efficiency and robustness are important requirements for the evaluation of a LCC object especially when there are large number of queries to make for the intermediate model and a large number of features in the LCC object or the composition profile is complicated.

Chapter 4

Definition and Classes of LCC Features

4.1 Introduction

The basic approach is to identify potential classes of LCC applications and for each class, identify features, which would be useful in design. For the purpose of allowing users to specify composition variation in the interior of a solid, an LCC feature is defined as follows:

An LCC feature is a construct composed of two major attributes:

(a) a generic parameterizable shape;

(b) a composition function defined over that shape.

LCC features as conceived here do not involve direct specification of higher level functional properties of a part (e.g., strength, wear), which are beyond the scope of this work. As shown in Figure 4-1, in terms of the data structure involved, an LCC feature is composed of two substructures/sub-features, one providing the representation of a generic shape, the other providing the representation of a composition profile. The arrow in Figure 4-1 illustrates that the composition sub-feature is applied to the domain of the geometric sub-feature whether the geometry is volumetric or in the form of surfaces. The block named "Interface methods" refers to the methods within LCC feature for user interaction. Therefore, an LCC feature can be viewed as primarily comprising two sub-features, for geometric shape and composition profile. The geometric sub-feature can be any standard geometric feature

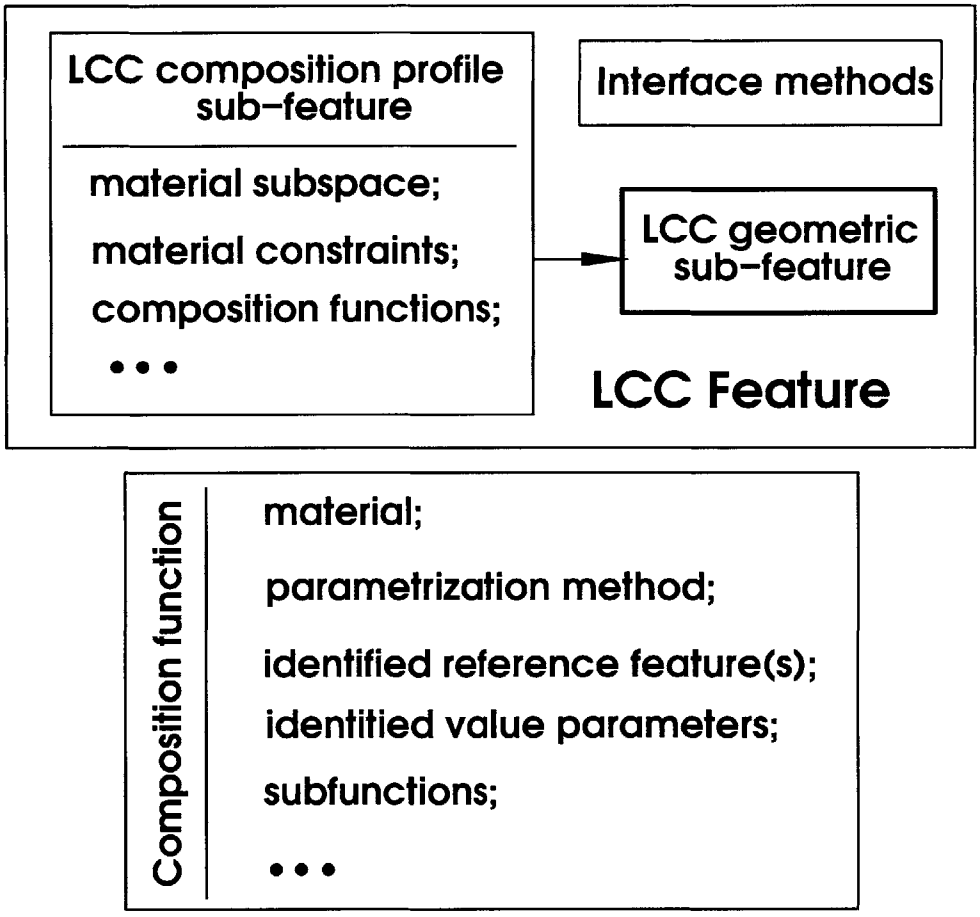


Figure 4-1: LCC feature class definition

or its extension to a general user-defined feature (UDF) [28], e.g. user-defined surface feature, volume feature, transition feature, pattern feature. Composition profile sub-feature has parameters such as material subspace and constraints on material composition. It also possesses attributes defined through composition functions. Material space is a catalog of materials available to the designer for LCC object design. Material subspace is a subset of the material space. Composition is the vector of volume fractions of each material defined over the material subspace and the generic shape of the feature. Composition function is the mapping function from the geometric sub-feature to the material subspace. Therefore, composition profile sub-feature is a dependent feature of the LCC geometric sub-feature. Composition constraints (design rules) are typically inequalities that specify e.g., what material composition or what gradient of material composition can be fabricated [18]. The procedural or declarative definitions in a manner analogous to conventional feature-based design are needed for the design by LCC features. LCC feature design should be tailored to both geometry and composition design intent.

LCC Feature Examples: A bimetalic sensor or actuator can be designed by defining the composition in a plane and extruding this composition along a line or sweeping it along a curve. A cylindrical Gradient Index Lens with composition gradient as a function of the distance from the axis of the cylinder and distance from the bottom face can be created by revolution of a 2D closed sketch, while the composition function is parametrized with respect to the axis and the 2D sketch entities. The impeller of Figure 4-2-(a), has fin features designed as bosses. Each fin has a composition which varies with height from the base so that it can have high wear resistance at the tip and ductility at the root. This feature is then replicated by patterning. In Figure 4-2-(b), a drug primitive has a composition as a function of distance to its axis, and a pattern of the drug is inserted to a pill.

Using the above examples as conceptual prototypes, creation methods for the two attributes of LCC features (generic shape and composition function) are discussed in Sections 4.2 to 4.5 and in Chapter 5.

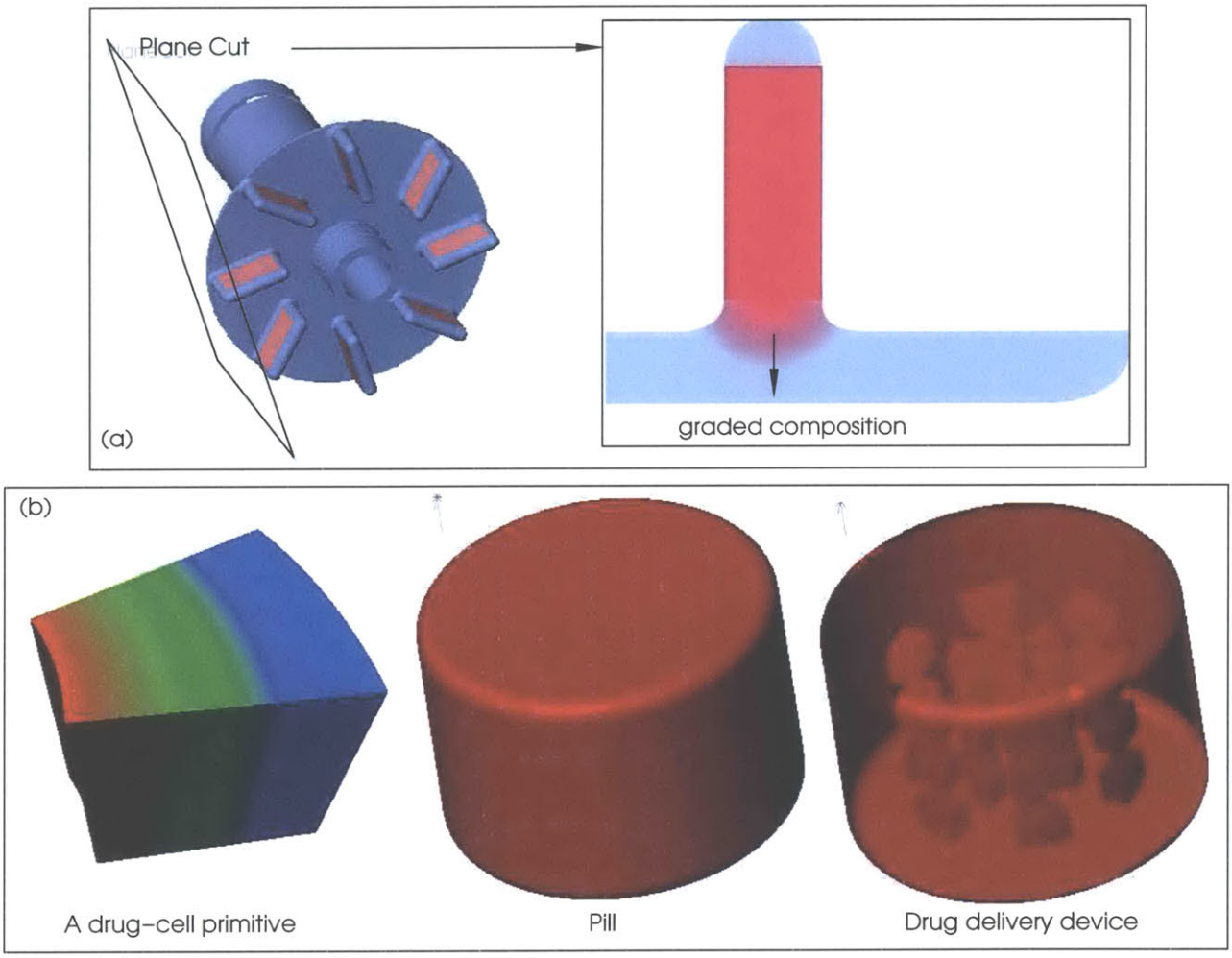


Figure 4-2: Conceptual illustration of parts created through LCC features

4.2 LCC volume feature

LCC volume feature is a graph of features in current FBD systems such as extrusion, boss, revolution, sweep, thicken, draft, holes, shell, surface pattern, surface fillet or all kinds of cuts [74]. Essentially it is a volume designed with sequences of form features (compound feature). The LCC volume feature is defined such that it can facilitate composition design for an LCC model of complicated compound features. According to Hoffmann and Arinyo [28], three primary form features are *generated features*, *modifying features* and *datum features*. In such a design paradigm, the generic shape is both an abstraction in terms of features and an evaluated geometry. For example, an extrusion feature is an example of generated feature, a surface fillet is a modifying feature. Datum features are the reference features. Under the user-defined feature design scheme, the feature structure is expressed with an acyclic directed graph where the nodes are form features and the edges are dependencies between them. A generic shape may frequently be created by alternative methods. For example, a cylinder can be created via either a revolution or an extrusion.

4.3 LCC surface feature

The generic shape of an LCC surface feature is the surface which is user-defined with a series of form features. For some applications graded composition may be desired for the volumetric domain bounded by a set of boundary surface features and composition variations need to be assigned on these surface features. An example is a smooth blending of composition between the composition values on different subsets of the boundary surface. The example of Figure 4-3 illustrates such a case. Here the top surface of the cylinder has a composition variation which is a function of distance to the axis of the cylinder, while the rest of the surfaces belong to the base-extrude feature and have compositions of a constant value. These are the two LCC surface features, i.e. the top surface and the rest of the surfaces of the cylinder. The domain of the volume is assigned a composition which is a blend of the LCC surfaces, and it is therefore an LCC volume feature. Another example involves applications that require users to define LCC feature shapes with half-space divisions. The half-spaces can be defined with either a plane or a curved surface and used as generic shapes

in LCC surface features.

4.4 LCC pattern feature

A *pattern feature* is a set of features arranged in a multi-dimensional array involving periodicity (eg. within cylindrical or rectangular coordinate frame). The generic shape of an LCC pattern is a group of volumes patterned from a seed volume. The material compositions are patterned from the seed LCC volume in the same way. Figure 4-2-(b) demonstrates the use of pattern feature to place drug cell primitives in a pill in a cylindrical lattice pattern. Figure 4-2-(a) also shows another example of a circular pattern feature.

4.5 LCC fillet feature

In conventional feature-based geometric modeling, transition features, such as fillets and chamfers, blend two or more surfaces. By extension, an LCC fillet feature blends the composition in a volumetric domain defined by transition surfaces and other user-defined surfaces which do not have to be conventional edge or corner blends. For example, for the root of the fins in Figure 4-2-(a), the specification of the transition volume can be constructed by surface cutting operations. The surfaces can be planes or curved surfaces. In order to edit the transition volume, the surfaces are defined with variable parameters. LCC fillet is a special type of the LCC volume feature in that the material compositions are smooth blends of the compositions in adjacent volumes. A blending method using Laplace's equation is developed and described in Chapter 5.

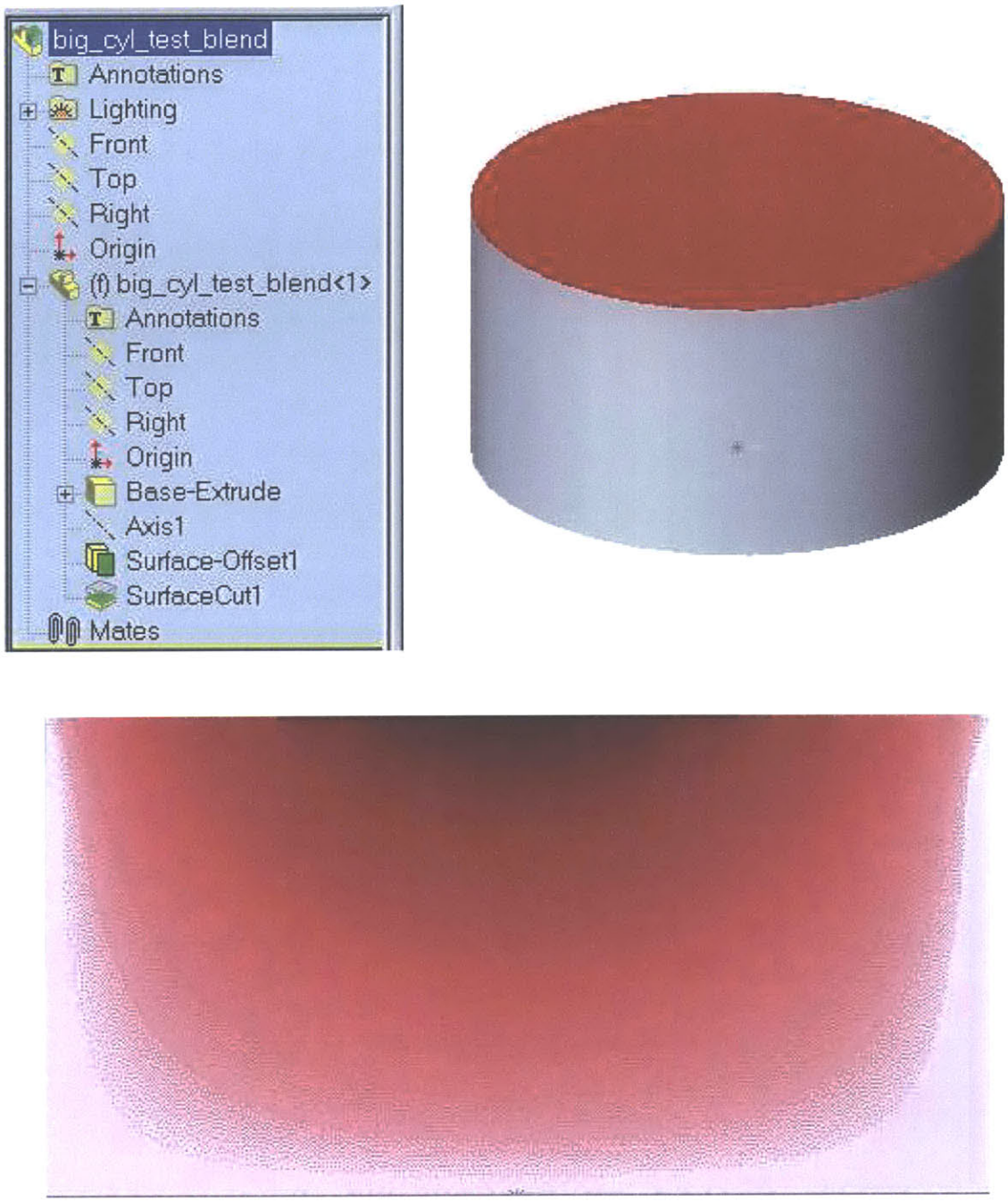


Figure 4-3: Top surface of the cylinder is an LCC surface with composition as function of distance to its axis

Chapter 5

Material Composition Functions

5.1 Introduction

A *Feature-Based Design* system provides a convenient environment for parametric design of composition because it carries rich information at different levels of abstraction. The scheme in this thesis is to specify the composition as a function of one or several variables. These variables are mapping from any interior point of user specified form features in the model to the material subspace, e.g., to specifying composition normal to a surface feature of a part. With such a scheme, when the reference form features are edited, the composition variables are either automatically reevaluated or the user is prompted to change a design for the composition function. The composition function of dependent LCC features can be designed from that of the parent LCC features. For example, general blending of two disjoint LCC features can be designed through a smoothing operation based on Laplace's equation. Such a design is based on user-defined features, and therefore it is also editable.

As shown in Figure 4-1, the composition function contains parameters such as the material it is related to, the parametrization method chosen by the user from the methods library, the identified reference features for parametrization, the identified value parameters and the subfunctions. The subfunctions include the functions that identify reference features from LCC geometric sub-feature or from user-defined feature through the GUI, functions that map feature into value parameters, analytical functions that map value parameters to composition ratio of the specified material, etc. For example, in case the feature uses

design according to distance from a reference feature, the subfunctions should contain the functions that identify and register the user-specified feature. The subfunctions may also contain the distance function from an input point to a reference feature. Finally, subfunctions may also contain the analytical functions that map the distance into a composition ratio of the specified material.

5.2 Distance function based design of composition function

Distance is chosen as the parameter for the design of composition because the distance function is a continuous function, distance as a concept is very intuitive, and distance to different form features captures a variety of parameters. For example, Cartesian coordinates are distances to three orthogonal planes.

5.2.1 Algorithm of digital distance transform

Distance to simple features such as axis and plane can be easily evaluated, but in the case that the minimum distances are evaluated to free-form surfaces, the computation of distance can be burdensome. Free-form surfaces are usually tessellated in advance of the minimum distance computation. In the case that the exact Euclidean distance value is needed, efficient distance computation developed by Liu *et al.* [42] can be used. The algorithm is based on preprocessing the geometry of the form features and the digital distance transform algorithm [9]. The method of preprocessing the geometry can be found in [42]. When the evaluation is done at a sufficiently fine resolution for visualization or printing, the exact distance can be approximated with the Euclidean digital distance. The square of the Euclidean digital distance is defined as:

$$\begin{aligned} S_{i,j,k} &= min\{d_t((i,j,k),(p,q,r))^2; f_{p,q,r} = 0, 1 \leq p \leq L, \\ &\quad 1 \leq q \leq M, 1 \leq r \leq N\}\ , \\ &= min\{(i-p)^2 + (j-q)^2 + (k-r)^2; \\ &\quad f_{p,q,r} = 0, 1 \leq p \leq L, \\ &\quad 1 \leq q \leq M, 1 \leq r \leq N\}\ , \end{aligned} \tag{5.1}$$

where L, M, N are the number of voxels in i, j, k respectively, and $f_{p,q,r}$ is any voxel that intersects the geometry of the reference features, whose Euclidean digital distance is therefore 0. The algorithm developed in [68] is one of the fastest in the literature. The following formulae summarize the basic algorithm:

1. $g_{ijk} = min\{(i-x)^2; f_{xjk} = 0, 1 \leq x \leq L\}$,
2. $h_{ijk} = min\{g_{iyk} + (j-y)^2; 1 \leq y \leq M\}$,
3. $s_{ijk} = min\{h_{ijz} + (k-z)^2; 1 \leq z \leq N\}$.

The efficiency is improved by reducing the search areas in step 2 and step 3 calculating h_{ijk} and s_{ijk}. For example, during step 2, the search is limited in $n_j = (g_{ijk} - g_{i(j-1)k} - 1)/2$ for each index j. This number is the intersection of the curves $f_1(n) = g_{ijk} + n^2$ and $f_2(n) = g_{i(j-1)k} + (n+1)^2$. The analysis for the Euclidean digital distance algorithm is given in a later section.

5.2.2 Single distance profile based design

This method computes composition in a single LCC feature as a single analytical function that uses distance as its independent variable. The distance is the minimum distance from an interior point of the LCC feature to a user selected form feature in the model. The analytical function is defined by the user and applied within a distance value limit supplied by the user. Beyond the distance limit, the value defaults to a constant which allows the function to be continuous at the limit distance. The form feature can be any generated feature, modifying feature or datum feature [28]. Examples in Figures 5-1 demonstrate such a design method using different form features. For example, in Figure 5-1-(a) an axis feature is used, while in Figure 5-1-(b) a user-defined surface and sweep feature are used.

5.2.3 Multi distance profile based design

The method and data structure

This method involves use of multiple distance function-based profiles to a single LCC feature simultaneously. Distance profiles are defined as shown in Figure 5-2-(a). With the upper

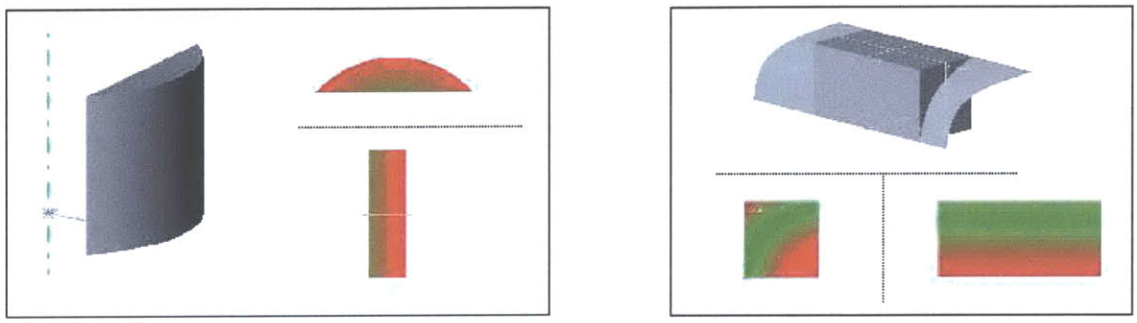


Figure 5-1: (a) Composition as function of distance to axis; (b) Composition as function of distance to different user defined surface

and lower distance limits, each distance profile is only applied within a subdomain of the LCC feature. The composition feature of LCC feature is represented by a queue of single distance based features. The last applied profile is at the tail of the queue. The profile at the front of the queue is the default constant value applied to the whole domain represented with Ω, which means for every point within Ω the material composition values will be the default constants. Thus the domain of the LCC feature can be represented as a binary subdivision tree with leaves which are quasi-disjoint subdomains as demonstrated in Figure 5-3. The root node of the tree has the domain of the whole volume of the LCC feature. Here in Figure 5-3 it is the whole bounded region of the rectangle. Each left node has the domain defined by the intersection of the parent node and the effective volume of the composition profile at that level. The right node has the domain defined by the difference. The effective volume of the composition profile is the point set such that from every point in the set the distance to the referenced feature is within the specific limit $d_l[d_u,\ d_p]$. For the example in Figure 5-2-(b), d_u is 0 for both profiles A and B. The composition at each left node (excluding the root) is defined as the weighted sum of that of the parent and the evaluated value of the composition profile at that level, which is expressed mathematically as

$$Comp_{left} = Comp_{parent} \cdot \frac{d_2}{d_1 + d_2} + Comp_{current\_profile\_in\_queue} \cdot \frac{d_1}{d_1 + d_2} \cdot \tag{5.2}$$

where d_1 represents the distance from a point located in the volume of the left node to the right node, d_2 represents the distance from a point located in the volume of the left node to the domain defined by the difference between the current profile and the parent. One special case is one in which when the domain of a left node completely encloses the domain of the profile, the distance d_2 is equal to zero. The composition at each right node inherits the composition of its parent node. Figure 5-2 demonstrates how this design method works. The rectangular shape represents a LCC feature, which has two distance-based composition profiles applied to two surface features A and B (A is the one with bold line) on the LCC feature. The whole region Ω of the LCC feature is then subdivided into 4 subdomains which are the leaf nodes in the tree. In Figure 5-2, the symbol RA represents the subset of Ω which satisfies that the minimum distance from any point within the subset to the feature A is bounded by the distance limits defined in the composition profile.

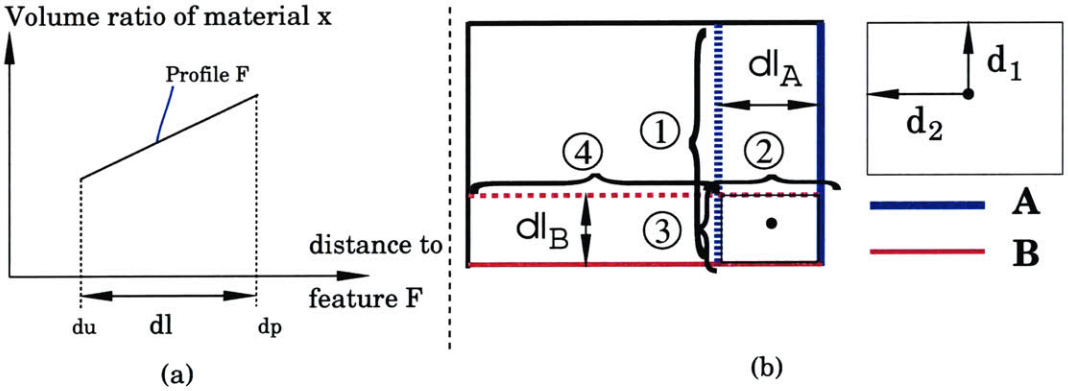


Figure 5-2: Design method for applying multiple distance based profiles

As in the design method of single distance-based profile, this thesis uses the digital distance transform for the evaluation of distance functions. A buffer is assigned for each distance profile for the distance transform. For efficient evaluation for visualization and printing, a buffer $\mathcal{B}$ is kept for the whole LCC feature. All these buffers are 3D arrays. With the use of the information in these buffers and the definition parameters such as the distance limits for the distance profiles, one can build up the above described binary tree. On each right node in the tree, a list of voxels is stored that define the interface boundaries between the domain of this right node and the domain of its sibling left node in the tree. And on each left node, a list of voxels is stored that define the interface boundaries between

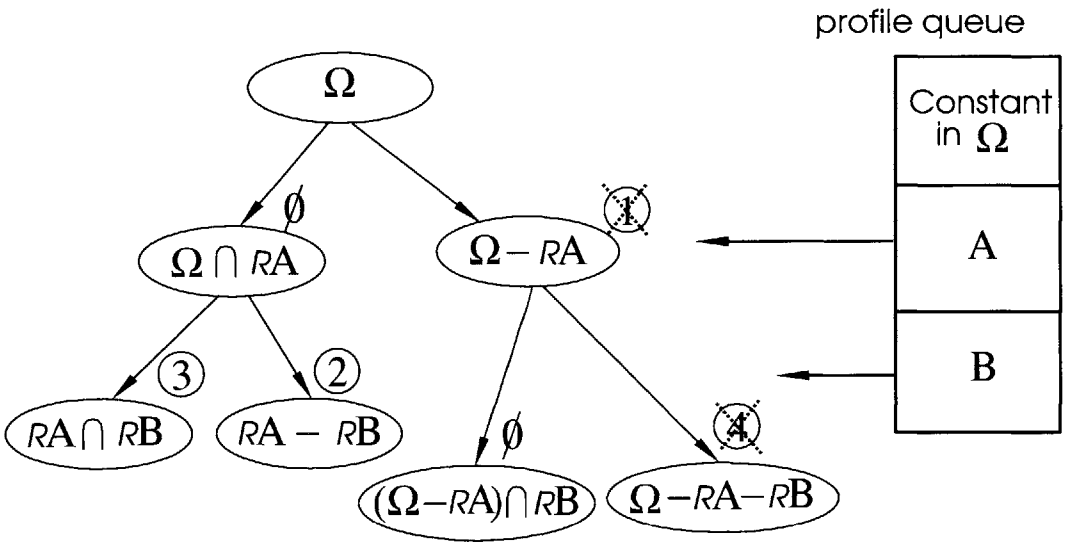


Figure 5-3: Subdivision tree underlying the multiple profile design method

the effective volume of the current distance profile and the domain of this left node. The algorithm for searching for these voxels is based on the Euclidean distance transform [68] buffers for each composition profile and the information stored for each voxel in buffer $\mathcal{B}$ that maps each voxel to a node in the binary tree. These two lists are used for the evaluation of d_1 and d_2. For the example in Figure 5-2, the voxel lists are stored as illustrated with numbers in circles in Figure 5-3. When the neighboring left node has an empty list, it is not necessary to store the list of the right node.

The algorithms

The multi-distance profile based design method is composed of the following major algorithms which are expressed in pseudo-code as follows:

Algorithm 1 Algorithm for preprocessing of multi-distance-based composition feature

1: initialize the binary tree;
2: initialize the dither matrix;
3: **for** each profile $\in$ the profile queue **do**
4: preprocess the binary tree based representation; ▷see Algorithm 2

Algorithm 2 Algorithm for updating the representation with a single distance-based composition profile

initialize the distance transform map with the associated geometric features;
do the Euclidean digital distance transform;
create/update the tree nodes based on the input single distance composition profile; ▷see Algorithm 3
process the dither profile lists that are associated with the tree nodes; ▷see Algorithm 4
evaluate the dither matrix after this operation for the related left node and right node; ▷see Algorithm 5

Algorithm 3 Algorithm for updating the tree nodes with a single distance-based composition profile

recursively updating the left child node with representation that saves the data about the intersection between the current domain of the node and the domain of the distance-based composition profile;
recursively updating the right child node with representation that saves the data about the difference of the domain of the distance-based composition profile from the current domain of the node;
if the current node is a leaf node **then**
  if there is intersection between the current domain of the node and the domain of the distance-based composition profile **then**
    add a left node;
  if the difference of the domain of the distance-based composition profile from the current domain of the node is not empty **then**
    add a right node;
  for each dither cell $\in$ the dither matrix **do**
    if the associated tree node for the dither cell is the current node **then**
      if the minimum distance from the center of the dither cell is within the designed limits d_l **then**
        associate the dither cell with the left child of the current node in the tree;
      else
        associate the dither cell with the right child of the crrent node in the tree.

Algorithm 4 Algorithm for processing the dither profile lists that are associated with the tree nodes

```
    if the left child of the current node is a leaf node and it is associated with the current
    input distance based composition profile then
      get the list of boundary cells that is stored with the left child node;
      get and set the bounding box of the intersection domain for use in computing of the
      nearest boundary cell ;
4:    set a walker pointing to the head of the list
      while the walker has not reached the end of the list do
        if the next cell to the walker is an interior cell within the domain defined by the
        intersection of the current node and the effective volume of the composition profile
        at that level then
          remove the cell from the list;
8:        walk to the next in the list;
        if the next cell to the walker is an interior cell within the domain defined by the
        difference between the current node and the effective volume of the composition
        profile at that level then
          move the cell to the right profile list;
          walk to the next;
12:     if the next cell to the walker belongs to neither of the above two categories then
          walk to the next;
    recursively process over the left child node;
    recursively process over the right child node;
```

Algorithm 5 Algorithm for evaluation of the dither matrix for the related left node and right node after a single distance-based composition profile is added

```
get the 3D grid system of the LCC object;
if the left child of the current node is a leaf node and its associated distance based
composition profile is the current input one then
  get the left list of boundary cells that is stored with the left child node;
  get the right list of boundary cells that is stored with the right child node;
  get the bounding box data of the list of boundary cells that are associated with the
  left child node;
  for each dither cell ∈ the bounding box do
    if the associated distance based composition profile with the cell is the left node of
    the current node then
      if the current node is not the most right node then
        calculate the distance d1 from the cell to the list on the right node;
        calculate the distance d2 from the cell to the list on the left node;
        if d1 + d2 equals zero then
          continue;
        calculate the material composition value only for the current distance profile
        based on the digital distance for this cell;
        evaluate the material composition value for this cell according to the formula
        as stated in eq. 5.2.;
      else
        calculate the material composition value only for the current distance profile
        based on the digital distance for this cell;
        set the material composition value;
if the right child of current node is a leaf node and its associated distance based
composition profile is the current input one then
  if current node is the root node of the tree then
    for each cell in the dither matrix do
      if the associated distance based composition profile with the cell is the right
      node of the current node then
        assign the default composition value;
recursively evaluate over the left child node;
recursively evaluate over the right child node;
```

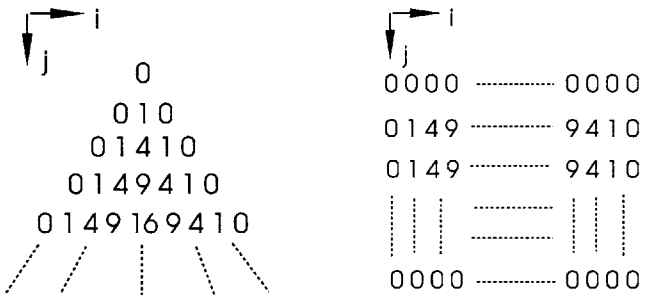


Figure 5-4: Cone and cube examples for analysis of EDT complexity

5.2.4 Complexity and accuracy analysis

Euclidean distance transform (EDT)

Complexity An analysis of the complexity of the EDT algorithm developed in [68] is summarized here. Given a 3D binary image with 0-voxels as the reference for distance calculation, the computation includes three parts, the time on computing g_{ijk} (T_1), the time on computing h_{ijk} (T_2) and the time on computing s_{ijk} (T_3). The time cost of the first step (T_1) is obviously $O(V)$, where $V = LMN$ is total number of voxels (see Section 5.2.1 for the definition of L, M, and N). The time cost of the second step is sensitive to the input, and therefore only the upper bound is given here. The algorithm for the second step will scan for each voxel, therefore, it will cost at least $O(V)$, plus for each j index, the algorithm will loop for at most $(g_{ijk} - g_{i(j-1)k} - 1)/2$ steps. Therefore, for each fixed pair of i and k, the worst estimate for the extra steps is $\sum_j^M (g_{ijk} - g_{i(j-1)k} - 1)/2$ which is equal to $(g_M - g_1 - M + 1)/2$. The value g_M is, in the worst case, the square of the half dimension of the image along i direction, and it is $L^2/4$. Given there are $L\ N$ number of pairs of i and k, the time cost for step 2 in the worst case is $O(V) + O(L^2\ L\ N) = O(V + L^3\ N)$. Similarly, the time cost for step 3 in the worst case is $O(V + L^3 M)$. If the dimension in each direction of the 3D image is the same, and denoted as N, then the worst time cost for steps 2 and 3 is of $O(N^4)$. The algorithm is sensitive to the input shape. In the following, two examples are given; one is a regular cone and the other is a cube.

Using these two examples, one can generate the input images after the first step. The images demonstrated in Figure 5-4 are the 2D images in $i - j$ planes that are in the middle of the two examples with respect to k axis. For the example of the cube, one can see that

in the step 2 computation, mostly the extra iteration of each j is zero, because mostly $g_j = g_{j-1}$. The extra iterations only happen to the voxel next to 0-voxels. The number of such voxels is of $O(N^2)$, and the extra iteration number for each such voxel is at most N. Therefore the extra iteration as a whole is of $O(N^3)$, and the time cost for step 2 is of $O(N^3)$, and similarly the time cost for step 3 is also of $O(N^3)$. Therefore, the time complexity of EDT on a cube is $O(N^3)$.

For the example of cone, consider the array of voxels in j direction with both i and k at the center of the planar face of the cone, the extra iteration of all the voxels in this array is of $O(N^2)$. For each array immediately surrounding the center array, the iteration is of $O[(N-1)^2]$. The number of such arrays that have the same number of iterations is linear in the radius relative to the axis. Therefore, the total number of extra iterations can be estimated by the following sum:

$$N^2 + c\,(N-1)^2 + \cdots + c\,N\,1^2,$$

which is equal to $N^2 + c\,\sum_{i=0} N - 2(N-i)(i+1)^2$. This sum can be simplified to $O(N^4)$ which is equal to $O(V^{4/3})$. one can see that a sphere is a shape that is somewhat in between these two shapes. Tests were conducted for three examples: a cube, a sphere and a tool part which is the example in Figure 8-4. The time performance curves on these models are shown in Figure 5-5. The time cost for the cube is linear in the volume, the time cost for the tool part is also almost linear in the volume, but slower in terms of slope, and the time cost for the sphere is about $O(V^{1.2})$.

Accuracy Given the definition of Euclidean digital distance, one can derive the maximum error of Euclidean digital distance as an approximation to the exact Euclidean distance between two points. For any two points located in voxels centered at X and Y respectively (Figure 5-6), with the assumption that the length of each voxel is δ, the square distance measured with Euclidean digital distance is

$$\overline{XY}^2 = [(|X_x - Y_x|)^2 + (|X_y - Y_y|)^2 + (|X_z - Y_z|)^2]\,\delta^2.$$

The maximum square distance between the two points is

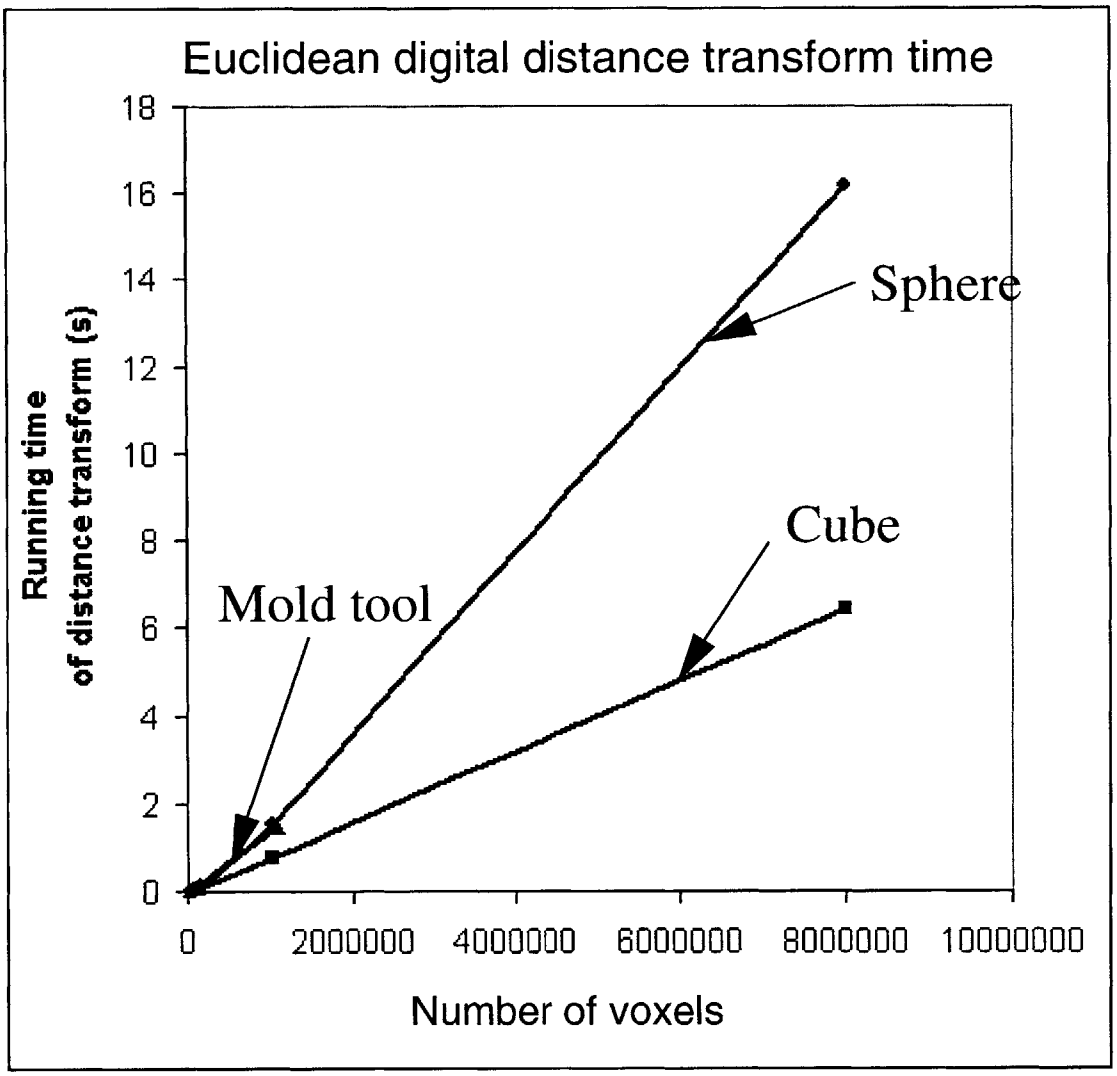


Figure 5-5: Time performance of Euclidean distance transform

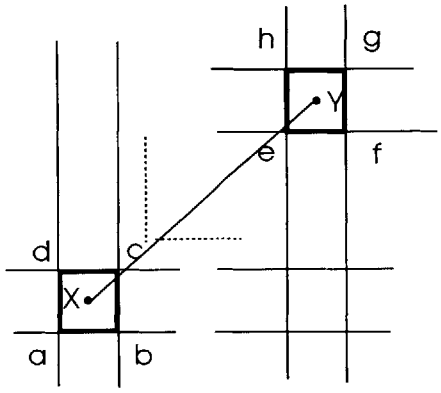


Figure 5-6: Accuracy of Euclidean digital distance

$$\begin{aligned}\overline{ag}^2 &= (\delta + \delta|X_x - Y_x|)^2 + (\delta + \delta|X_y - Y_y|)^2 + \\ &\quad (\delta + \delta|X_z - Y_z|)^2 , \\ &= 3\delta^2 + 2\delta^2(|X_x - Y_x| + |X_y - Y_y| + \\ &\quad |X_z - Y_z|) + \overline{XY}^2 . \end{aligned} \tag{5.3}$$

Therefore the maximum error in terms of the square distance is:

$$\overline{ag}^2 - \overline{XY}^2 = 3\delta^2 + 2\delta^2(|X_x - Y_x| + |X_y - Y_y| + |X_z - Y_z|)$$

And

$$\begin{aligned}&|X_x - Y_x| + |X_y - Y_y| + |X_z - Y_z| \leq \\ &\sqrt{3} \cdot \sqrt{(|X_x - Y_x|)^2 + (|X_y - Y_y|)^2 + (|X_z - Y_z|)^2} , \\ &\qquad = \sqrt{3}\overline{XY}/\delta \qquad . \end{aligned} \tag{5.4}$$

Therefore, the maximum error in terms of square distance is $3\delta^2 + 2\sqrt{3}\delta\overline{XY}$

Complexity analysis for the multi-distance profile based design

The complexity of the algorithms for the binary tree represented multi-distance profile based design feature is a function of the number of profiles in the queue. Here this number is denoted by the symbol N_q. If the tree is complete which means that the domain introduced

by a newly added profile intersects almost all the domain of the current leaf nodes of the tree, then the number of nodes of the tree is of $O(2^{N_q})$. As analyzed before, the complexity of Euclidean digital distance transform is $O(V)$. Therefore the worst case time complexity of the algorithms is $O(2^{N_q}V)$, where V is the number of voxels in the digital distance transform map. In practical design, especially when the reference features of the distance profiles are separate surface features of the solids, the tree is very unlikely to be full. It is conceivable that the newly introduced domain by the added profile intersects only constant number of the node domains in the tree. In this scenario, the number of nodes that need to be traversed in the algorithms is of $O(N_q)$. Then the time complexity of the algorithms is $O(N_qV)$.

5.3 Laplace blending based design

The use of Laplace's equation to compute the blending of the material composition from the boundary conditions is motivated by its use in surface design. Laplace's equation has been used extensively in smooth surface design with few parameters, as in Bloor and Wilson [8]. Although only the constant coefficient Laplace equation based blending is presented here, the same method can be applied to more general elliptic partial differential equation problems. Qian and Dutta [61] presented a related diffusion-based design method for heterogeneous material turbine blades.

5.3.1 Blending function algorithm

Setting of the boundary conditions

Users can intuitively assign boundary conditions for blending by using the LCC surface features. The procedure is to select one or several surface features in the model and assign some composition design profile to apply on them, then select the domain to blend the LCC surface features, and the system will set the boundary conditions automatically. In the case that the blending is between several LCC volumes, the boundary conditions (boundary composition value or its normal directional derivative across the intervening boundary surface) are derived from those LCC volumes automatically. If all the involved geometric features

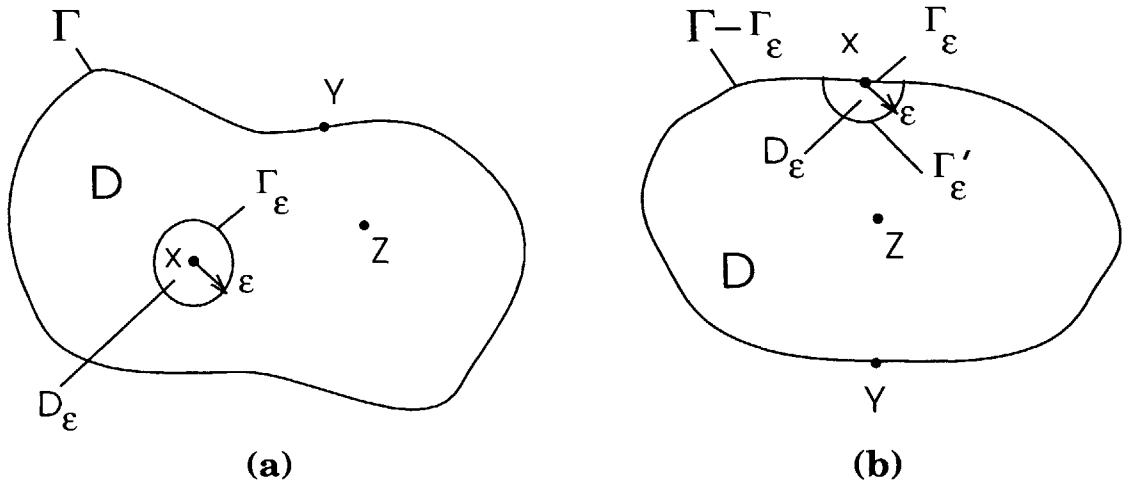


Figure 5-7: Domain type: (a) Point $\mathbf{x}$ is in the interior of the domain D; (b) Point $\mathbf{x}$ is on the boundary Γ.

are parameterizable, the users can edit the parameters affecting the Laplace equation-based blending.

Solving Laplace's equation with the Boundary Element Method

This thesis employs the Boundary Element Method (BEM) [56, 10], which utilizes the second form of Green's theorem [56] to express the potential function in the domain by an integral representation involving the potential function value and its normal directional derivative on the boundary and the fundamental solution of Laplace's equation.

If ϕ and ψ are scalar functions of position defined on a region D bounded by a closed surface Γ (see Figure 5-7-(a)), the second form of Green's theorem [56] is given by

$$\begin{aligned} &\int_D (\phi(\mathbf{z})\nabla^2\psi(\mathbf{x},\mathbf{z}) - \psi(\mathbf{x},\mathbf{y})\nabla^2\phi(\mathbf{z}))dv(\mathbf{z}) \\ &= \int_\Gamma \left(\phi(\mathbf{y})\frac{\partial\psi}{\partial n}(\mathbf{x},\mathbf{y}) - \psi(\mathbf{x},\mathbf{y})\frac{\partial\phi}{\partial n}(\mathbf{y})\right) ds(\mathbf{y}) \,, \end{aligned} \tag{5.5}$$

where $\mathbf{x}, \mathbf{z} \in D$ and $\mathbf{y} \in \Gamma$.

If we assume that the function ϕ corresponds to the variable in question and ψ corre-

sponds to the fundamental solution of Laplace's equation, they satisfy

$$\nabla^2\phi(\mathbf{z}) = 0 \qquad \mathbf{z} \in D\ , \tag{5.6}$$

$$\nabla^2\psi(\mathbf{x},\mathbf{z}) + \Delta(\mathbf{x},\mathbf{z}) = 0, \quad \Delta(\mathbf{x},\mathbf{z}) = 0, \mathbf{x} \neq \mathbf{z}$$

$$\mathbf{x},\mathbf{z} \in D\ , \tag{5.7}$$

where $\Delta(\mathbf{x},\mathbf{z})$ is Dirac's delta function, which plays a role of a unit source applied at point $\mathbf{x}$. The solution to (5.7) is given by

$$\psi = \frac{1}{4\pi r}\ , \tag{5.8}$$

where r is the Euclidean distance between the unit source at $\mathbf{x}$ and the point $\mathbf{z}$ that are of interest, i.e. $r = |\mathbf{x} - \mathbf{z}|$. Since ψ has a singularity at $\mathbf{x}$, we need to isolate this point to avoid integration through a singularity. As in Figure 5-7-(a), if we enclose the domain D_ϵ, which includes point $\mathbf{x}$ by a sphere Γ_ϵ of radius ϵ, the new integration domain will become $D - D_\epsilon$ with boundary $\Gamma + \Gamma_\epsilon$. The result is obtained by considering the limit as ϵ tends to zero. Consequently the second form of Green's theorem in equation (5.5) can be rewritten as

$$\begin{aligned}
&\lim_{\epsilon\to 0}\int_{D-D_\epsilon} \phi(\mathbf{z})\nabla^2\psi(\mathbf{x},\mathbf{z})dv(\mathbf{z}) - \\
&\lim_{\epsilon\to 0}\int_{D-D_\epsilon} \psi(\mathbf{x},\mathbf{z})\nabla^2\phi(\mathbf{z})dv(\mathbf{z}) \\
&= \int_\Gamma \left(\phi(\mathbf{y})\frac{\partial\psi}{\partial n}(\mathbf{x},\mathbf{y}) - \psi(\mathbf{x},\mathbf{y})\frac{\partial\phi}{\partial n}(\mathbf{y})\right) ds(\mathbf{y}) + \\
&\lim_{\epsilon\to 0}\int_{\Gamma_\epsilon} \left(\phi(\mathbf{y})\frac{\partial\psi}{\partial n}(\mathbf{x},\mathbf{y}) - \psi(\mathbf{x},\mathbf{y})\frac{\partial\phi}{\partial n}(\mathbf{y})\right) ds(\mathbf{y})\ ,
\end{aligned} \tag{5.9}$$

where $\mathbf{z} \in D - D_\epsilon$, $\mathbf{x} \in D_\epsilon$, $\mathbf{y} \in \Gamma$ or Γ_ϵ. Using equation (5.7), the left hand side of equation (5.9) is zero. The limit of the first part of the singularity integral of the right hand side converges to $\phi(\mathbf{x})$, while the second tends to zero. The limits of these singular integrals exist independent of how ϵ goes to zero, and the singularity is said to be *weak*. Rewriting

of (5.9) leads us to the fundamental equation of the BEM:

$$\phi(\mathbf{x}) = \int_{\Gamma} \left(\psi(\mathbf{x},\mathbf{y}) \frac{\partial \phi}{\partial n}(\mathbf{y}) - \phi(\mathbf{y}) \frac{\partial \psi}{\partial n}(\mathbf{x},\mathbf{y}) \right) ds(\mathbf{y}) \ , \tag{5.10}$$

where $\mathbf{x} \in D$ and $\mathbf{y} \in \Gamma$. This basic equation implies us that at any point $\mathbf{x}$ inside the domain D, the potential function $\phi(\mathbf{x})$ can be obtained from the potential $\phi(\mathbf{y})$ and its derivative $\frac{\partial \phi}{\partial n}(\mathbf{y})$ on the boundary. However $\phi(\mathbf{y})$, $\frac{\partial \phi}{\partial n}(\mathbf{y})$ are initially not known and need to be computed, which is the main task of BEM.

To obtain the unknowns $\phi(\mathbf{y})$, $\frac{\partial \phi}{\partial n}(\mathbf{y})$ we start from (5.5) again, but this time $\mathbf{x}$ is on the boundary instead of being inside the domain. Similar to the case when $\mathbf{x}$ is inside the domain, the left hand side is zero and we have

$$\begin{aligned} 0 = \lim_{\epsilon \to 0} \int_{\Gamma - \Gamma_\epsilon} \left(\phi(\mathbf{y}) \frac{\partial \psi}{\partial n}(\mathbf{x},\mathbf{y}) - \psi(\mathbf{x},\mathbf{y}) \frac{\partial \phi}{\partial n}(\mathbf{y}) \right) ds(\mathbf{y}) + \\ \lim_{\epsilon \to 0} \int_{\Gamma'_\epsilon} \left(\phi(\mathbf{y}) \frac{\partial \psi}{\partial n}(\mathbf{x},\mathbf{y}) - \psi(\mathbf{x},\mathbf{y}) \frac{\partial \phi}{\partial n}(\mathbf{y}) \right) ds(\mathbf{y}) \ , \end{aligned} \tag{5.11}$$

where $\mathbf{x} \in \Gamma_\epsilon$ and $\mathbf{y} \in \Gamma - \Gamma_\epsilon$ or Γ'_ϵ (see Figure 5-7-(b)). The limit of the first part of the second singularity integral of the right hand side converges to $\frac{\alpha(\mathbf{x})}{2\pi}\phi(\mathbf{x})$, where $\alpha(\mathbf{x})$ is the internal angle at point $\mathbf{x}$, while the second part tends to zero. For a smooth boundary $\alpha(\mathbf{x})$ is π. The first part of the first singularity integral in (5.11) can be evaluated in the sense of Cauchy Principal Value, as the limit is undefined unless ϵ satisfies certain conditions as it approaches zero, while the second part is a weak singularity and the limit exists independently of how ϵ approaches zero. Equation (5.11) therefore reduces to

$$\begin{aligned} \frac{\alpha(\mathbf{x})}{2\pi}\phi(\mathbf{x}) + \mathrm{P}\int_{\Gamma} \phi(\mathbf{y}) \frac{\partial \psi}{\partial n}(\mathbf{x},\mathbf{y}) ds(\mathbf{y}) = \\ \int_{\Gamma} \psi(\mathbf{x},\mathbf{y}) \frac{\partial \phi}{\partial n}(\mathbf{y}) ds(\mathbf{y}) \ , \end{aligned} \tag{5.12}$$

where $\mathbf{x}, \mathbf{y} \in \Gamma$ and the symbol P denotes a principal value of an integral.

The boundary of the region Γ is discretized by triangular elements $\Gamma_T = \sum_{k=1}^{n} \Gamma_{T_k}$. The collocation points of the boundary integral equation are located at the centroids of the elements. We need to supply either the composition value ϕ or the normal derivative of ϕ,

i.e.$\frac{\partial\phi}{\partial n}$, at each collocation point. Since the collocation points are taken at the centroids of the elements, the boundary is always smooth there, therefore $\alpha(\mathbf{x}) = \pi$, and hence (5.12) at the centroids of the elements takes the form:

$$\frac{1}{2}\phi(\mathbf{x}) + \mathrm{P}\int_{\Gamma_T} \phi(\mathbf{y})\frac{\partial\psi}{\partial n}(\mathbf{x},\mathbf{y})ds(\mathbf{y}) = \int_{\Gamma_T} \psi(\mathbf{x},\mathbf{y})\frac{\partial\phi}{\partial n}(\mathbf{y})ds(\mathbf{y}) \ , \tag{5.13}$$

where $\mathbf{x}, \mathbf{y} \in \Gamma_T$. At the collocation point $\mathbf{x} = \mathbf{x}_j$, Equation (5.13) becomes:

$$\frac{1}{2}\phi(\mathbf{x}_j) + \sum_{k=1}^{n} \phi(\mathbf{x}_k)\mathrm{P}\int_{\Gamma_{T_k}} \frac{\partial\psi}{\partial n}(\mathbf{x}_j,\mathbf{y})ds(\mathbf{y}) = \sum_{k=1}^{n} \frac{\partial\phi}{\partial n}(\mathbf{x}_k)\int_{\Gamma_{T_k}} \psi(\mathbf{x}_j,\mathbf{y})ds(\mathbf{y}) \ , \tag{5.14}$$

where $\mathbf{y} \in \Gamma_{T_k}$ and $\mathbf{x}_k$ represents the k-th nodal point. Equation (5.14) can now be written as the linear system

$$Ax = b \ , \tag{5.15}$$

where A is a non-singular non-symmetric full $n \times n$ matrix, b is a vector of length n, and x represents the unknown values of $\phi(\mathbf{y})$ or $\frac{\partial\phi}{\partial n}(\mathbf{y})$.

Design rule related to Laplace's equation based blending

Design of a multi-material 3D solid using Laplace's equation based blending is done by assigning the composition of each material according to the equation and the composition of the last material is set equal to $1 - \sum$ Composition volume fractions of the rest of the materials.

The solution of the blending is reduced to the solution of Laplace's equation over a closed bounded region with boundary value conditions. When all of the subsets of the boundary are assigned values of the function, the resulting problem is a Dirichlet problem. In the case of volume material filleting, the system allows the user not to specify the boundary values for some subsets of the boundary and the system will assume the condition for those

subsets is such that the normal derivative of the boundary value is equal to 0. Such a case is a special case of the Neumann problem.

The proof for the following proposition is given for the above two cases:

Proposition 1: If the boundary values are within the range of $[0, 1]$, then the solution for any point within the domain is within the range of $[0, 1]$ and the sum of the composition of all the materials designed with Laplace's equation is equal to 1.

The proof of this proposition employs the following Theorem.

Theorem [37]: Assume that Ω is a bounded, open and connected set in $\Re^n$. Let $u \in C^2(\Omega) \cap C^0(\bar{\Omega})$, and let $\Delta u \geq 0$ in Ω. Then

$$\max_{\bar{\Omega}} u = \max_{\partial\Omega} u$$

Notes on the Theorem: Notice that a continuous function u assumes its maximum somewhere in the closed and bounded set $\bar{\Omega}$. The above formula asserts that u assumes its maximum certainly on the boundary of Ω, possibly also in Ω [37]. From the above theorem, the maximum and minimum principles can be derived such that the maximum and minimum values of a harmonic function ϕ occurs on the boundary. If the maximum or minimum is attained in the interior, then the solution is identically constant.

Proof of Proposition 1:

In the case of the Dirichlet problem, if the boundary values are specified within the range $[0, 1]$, then the value of any interior point should be in the range of $[0, 1]$. For the composition design for multiple materials, if the boundary values for different materials sum to 1, given the linearity of Laplace's equation, the sum of values for different materials everywhere in Ω satisfies Laplace equation with boundary value of 1 everywhere on $\partial\Omega$. The solution to that equation therefore is $\phi = 1$.

In the case of the special Neumann problem, refering to Figure 5-8, we can prove that the maximum/minimum value for ϕ_1 or ϕ_2 does not happen only on the subsets that have zero normal derivative by the method of contradiction. If that scenario is true, then it contradicts the zero normal derivative condition on those subsets. Therefore, the boundary values on the subsets that have zero normal derivative should be within the maximum and

minimum range set by the boundary value profile A and B. Since A and B are in the range $[0, 1]$, then the boundary values everywhere on $\partial\Omega$ should be in the range $[0, 1]$, therefore the value of any interior point should be in the range of $[0, 1]$ as well. Similar to the proof for the Dirichlet problem, given the linearity of Laplace equation, the sum of values for different materials everywhere in Ω satisfies Laplace equation with boundary value of 1 on some subset of $\partial\Omega$ and has zero normal derivative on the remaining subset of the boundary. The solution to that equation should also be $\phi = 1$. In this way, the above proposition is

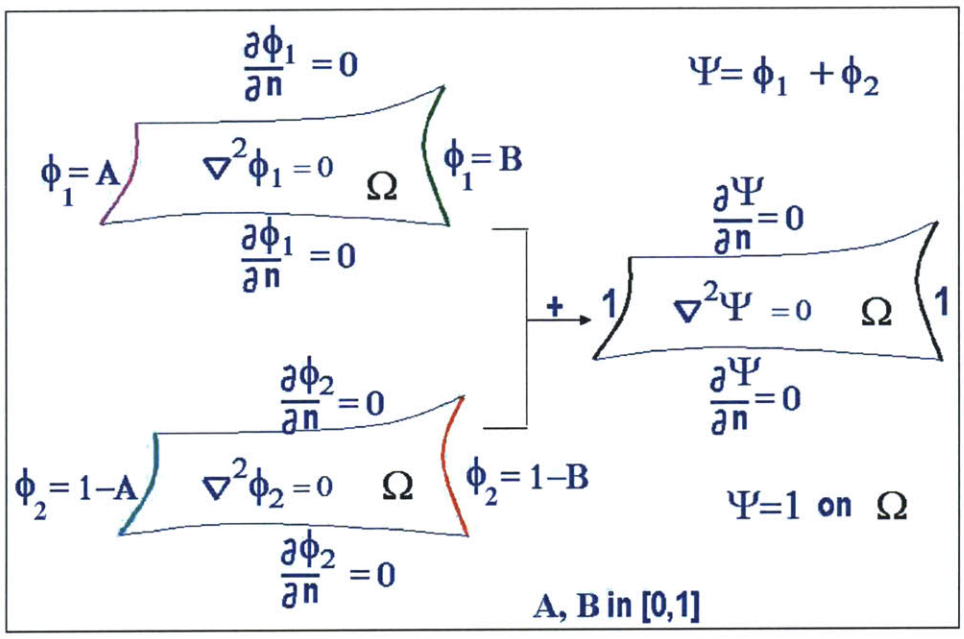


Figure 5-8: Proof for the design rule of blending with Laplace equation

proved.

Solution methods for the linear system

The drawback of the BEM compared with the Finite Element Method (FEM) is that the matrix A is non-symmetric and fully occupied [10, 56]. Traditionally, these linear systems were solved by the direct method, such as the Gauss elimination method due to its robustness. However its computational cost is proportional to n^3, where n is the size of the matrix, and such cost becomes prohibitive when n is large. Recently, a number of efficient iterative methods were developed and are gaining in popularity. Among these newly devel-

oped iterative methods, this thesis adopted the Generalized Minimum Residual (GMRES) method which is useful for general non-symmetric matrices [40, 63]. If we denote the initial approximation by x_o, the corresponding residual of (5.15) can be written as $r_o = b - Ax_o$. Also let us denote the Krylov subspace of dimension m by

$$K_m(A, r_o) = span\{r_o, Ar_o, A^2 r_o, \cdots, A^{m-1} r_o\} , \tag{5.16}$$

where

$$A^k = \prod_{i=1}^{k} A^i$$

. In GMRES the solution of (5.15) is approximated by

$$x_m = x_o + V_m y \tag{5.17}$$

where V_m is an orthonormal basis for the Krylov subspace of dimension m, and y is a vector of length m (typically m is small compared with n). The orthonormal basis V_m is constructed through Arnoldi's procedure which uses the Modified Gram-Schmidt orthogonalization algorithm [63]. The vector y is determined so that the norm of the residual $r_m = b - Ax_m$ is minimized. It is guaranteed that the GMRES algorithm converges in at most n steps in exact arithmetic, however the algorithm becomes impractical when m is large because of the memory growth of the orthonormal basis V_m. The restarted GMRES overcomes this storage limitation by restarting the iteration after a chosen number of iterations.

When the condition number of the matrix A is large, the GMRES may suffer from a slow rate of convergence and low accuracy. To overcome the efficiency and accuracy problems of iterative methods, *preconditioning* is introduced, where the linear system (5.15) is transformed into one that has the same solution, but has a smaller condition number. If a preconditioning matrix M approximates the matrix A in some way, then the matrix $M^{-1}A$ would be close to the identity matrix, and hence may have a smaller condition number. If

we pre-multiply equation (5.15) with M^{-1}, the transformed linear system becomes

$$M^{-1}Ax = M^{-1}b\,. \tag{5.18}$$

This system has the same solution as that of (5.15), and an iterative method for its solution may perform better in efficiency and accuracy. The preconditioning done in this way is called *left conditioning*. Of course, one can set:

$$AM^{-1}y = b\,, \tag{5.19}$$

which is first solved for y, and then for the solution x

$$x = M^{-1}y\,. \tag{5.20}$$

In this case matrix M is called the *right preconditioner*. In this work, a symmetric successive overrelaxation (SSOR) left preconditioner [63, 40] was employed in which

$$M = (D + \omega L)D^{-1}(D + \omega U)\,, \tag{5.21}$$

where D, L, U are the diagonal, strict lower triangular, strict upper triangular matrices of A, respectively, and ω is a scalar parameter taking the value between 0 and 1. One can also express the SSOR preconditioner as

$$\begin{aligned} M &= (I + \omega LD^{-1})(D + \omega U)\,, \\ &= (D + \omega L)(I + \omega D^{-1}U)\,. \end{aligned} \tag{5.22}$$

In the actual implementation, M is never inverted and the system $Mr = b - Ax$ is solved for the residual r by taking the advantage of triangular decomposition. If

$$M = (D + \omega L)(I + \omega D^{-1}U),$$

then the algorithm first solves the lower triangular system

$$(D + \omega L)y = b - Ax \ , \tag{5.23}$$

for y and then solves the upper triangular system

$$(I + \omega D^{-1}U)r = y \ , \tag{5.24}$$

for r. In the implementation, the value $\omega = 0.5$ is used. The pseudocode for the GMRES algorithm with left preconditioner is given in Appendix A.2 adapted from [63].

5.3.2 Complexity and convergence analysis

Complexity

From the above description on the BEM method, one can see that the computation time of the method is $T = T_1 + T_2$, where T_1 and T_2 are the times spent on setting up and solving the linear equation system, respectively. Here the analysis omits the time on meshing the boundary into planar panels. If the number of panels is equal to n, which is equal to the dimension of the matrix A for the equation system (5.15), then we have n^2 coefficients in the matrix for evaluation, and the time for each coefficient is constant, therefore the time T_1 is asymptotically of $O(n^2)$. The algorithm of the left-preconditioned GMRES is given in Appendix A.2. This thesis has used the GMRES algorithm developed by Frayssé *et al.* [25]. The performance of the GMRES algorithm depends on both the number of boundary elements (n) and the number of iterations m of GMRES. Line 1 of the Algorithm consists of matrix-vector multiplication, forward substitution (5.23) and back substitution (5.24) of a triangular system. Since matrix-vector multiplication takes n^2 multiplications and additions, and each substitution takes $\frac{n^2}{2}$ arithmetic operations, in total Line 1 costs $2n^2$. Line 4 also involves matrix-vector multiplication, forward substitution and back substitution of a triangular system for each j step, resulting in $2mn^2$ arithmetic operations for m iterations. Line 6 is an inner vector product and Line 7 is a constant-vector multiplication and a vector subtraction where both lines are inside the nested for-loops. Therefore the two lines take $m(m+1)n$ arithmetic operations. Line 13 involves transformation of a Hessenberg matrix

into the upper triangular form by using the Givens rotation matrix [40] which costs order $\frac{m(m+1)(m-1)}{3}$ operations, and solving the resulting triangular system which can be solved in order $\frac{m(m+1)}{2}$ operations. Consequently the GMRES algorithm performance time T_2 is of $O(mn^2 + m^2n + m^3)$. In conclusion, the time cost on solving the blending problem with Laplace's equation is $O(n^2) + O(mn^2 + m^2n + m^3)$. Because in most cases m is small compared with n, the time complexity is almost $O(n^2)$. Figure 5-9 gives the experimental result on a model called "Sample_split", which confirms our analysis. The computation time with the LU decomposition method, which is one of the direct methods for solving a linear equation system, is $O(n^3)$, while the computation time with GMRES is $O(n^2)$.

Convergence

Studies of BEM methods have shown that the convergence rate of the constant panel method is of $O(n^{-2})$ [80]. Tests were conducted on the example "Cube" with boundary conditions such as with one face out of the 6 faces at constant value 1 and the parallel one to it set at value 0.5 and the rest of the 4 faces at constant 0. Compared with the theoretical solution [14], the result is shown in Figure 5-10 which confirms the quadratic convergence. Figure 5-11 shows the convergence test on the example in Figure 8-7. With the number of panels equal to 1000, the relative error of the solution is already less than 2% when the linear system solution is precisely computed. The tests also include the convergence of the GMRES method with and without left pre-conditioning. Figure 5-13 shows the result on a "Mug" example shown in Figure 5-12. The inner surfaces of the "Mug" is designed as a LCC surface with material composition of constant value 0. The outer surface of the "Mug" is designed as a LCC surface with constant composition value 1. And the composition for the solid volume is a Laplace equation-based blending. The matrix generated from this example has a large condition number (4.1×10^7) compared with other examples, such as the condition number 7.6×10^4 for the example of Figure 8-8-(b). Without preconditioning, the solution will not converge if a solution better than the resolution of 10^{-3} is desired. With preconditioning, the convergence rate of GMRES is significantly improved. For the example of Figure 8-8-(b), it was tested for different densities of discretizations, and Figure 5-14 shows a significant reduction in the number of iterations when solving the system with GMRES method using left pre-conditioning.

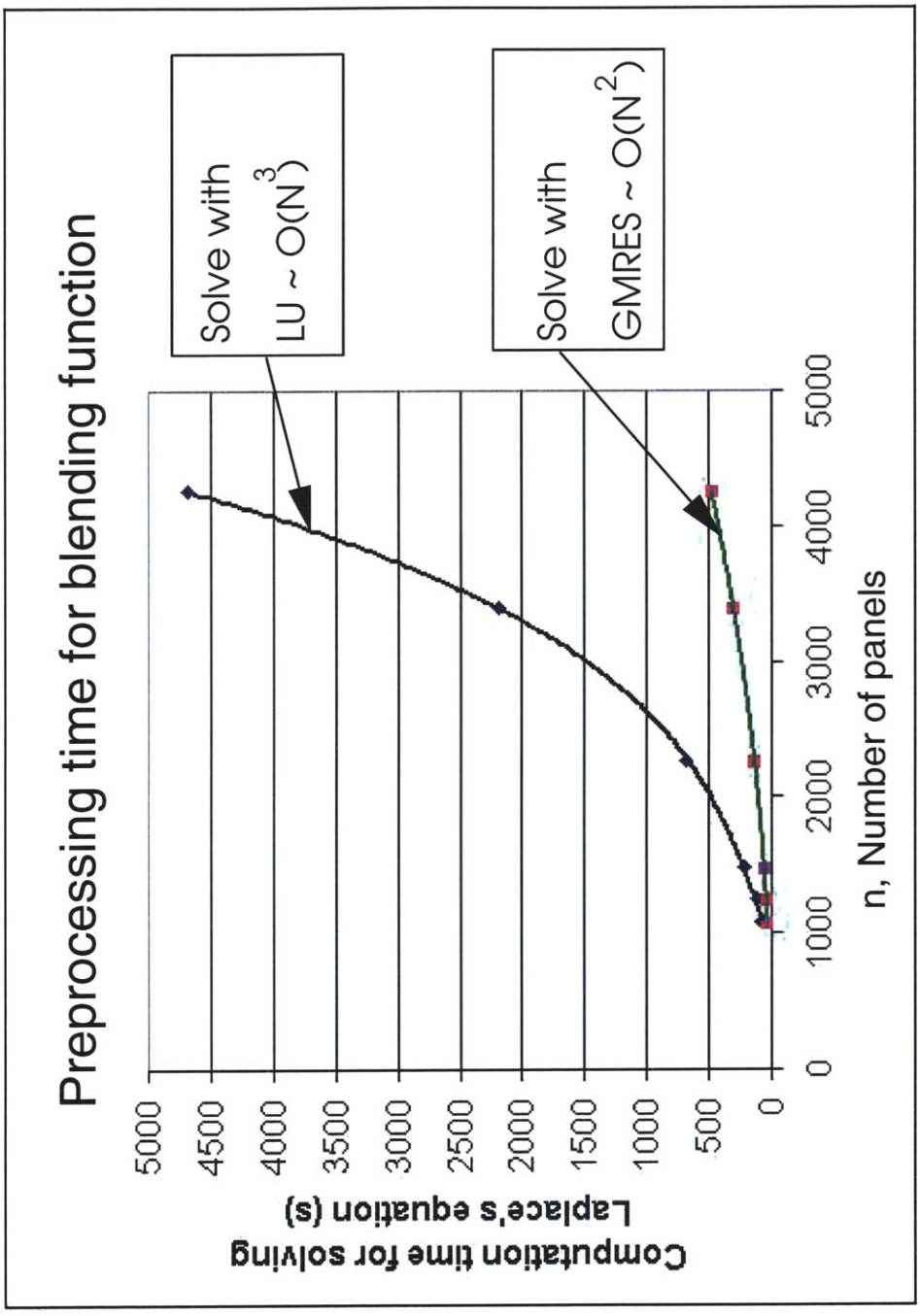


Figure 5-9: Computation time of the blending function on example "Sample_split"

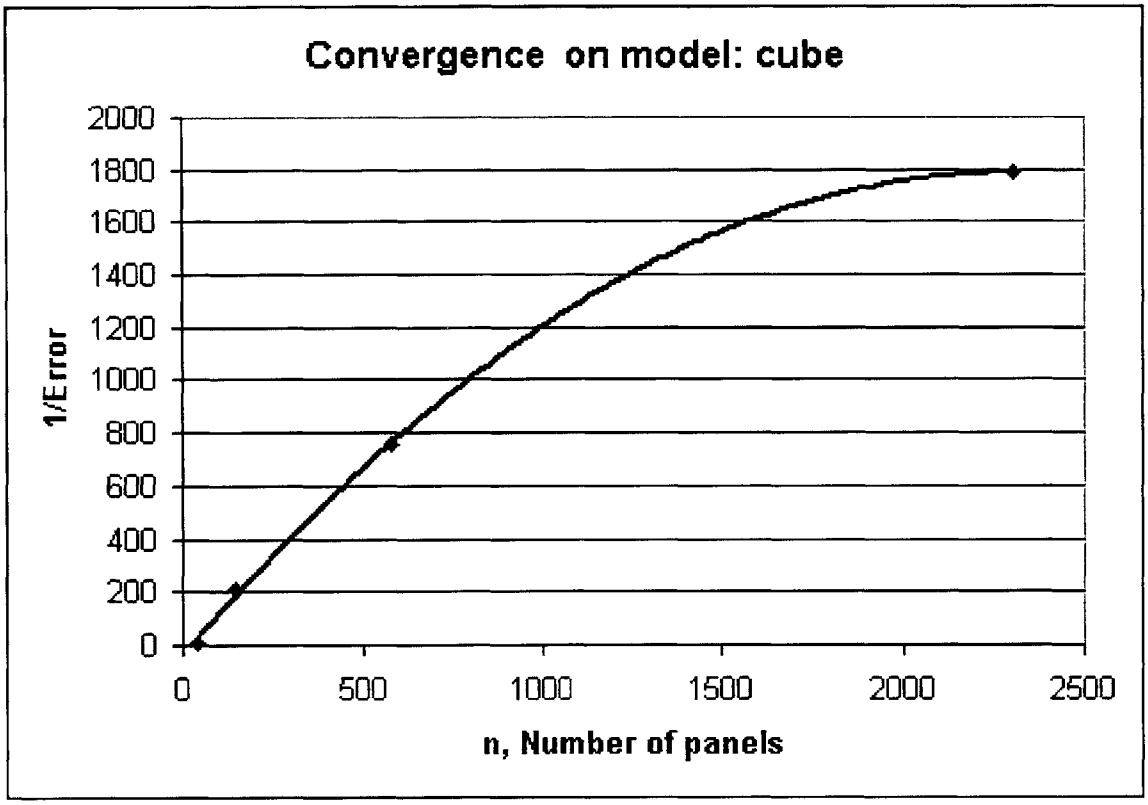


Figure 5-10: Convergence on example "Cube"

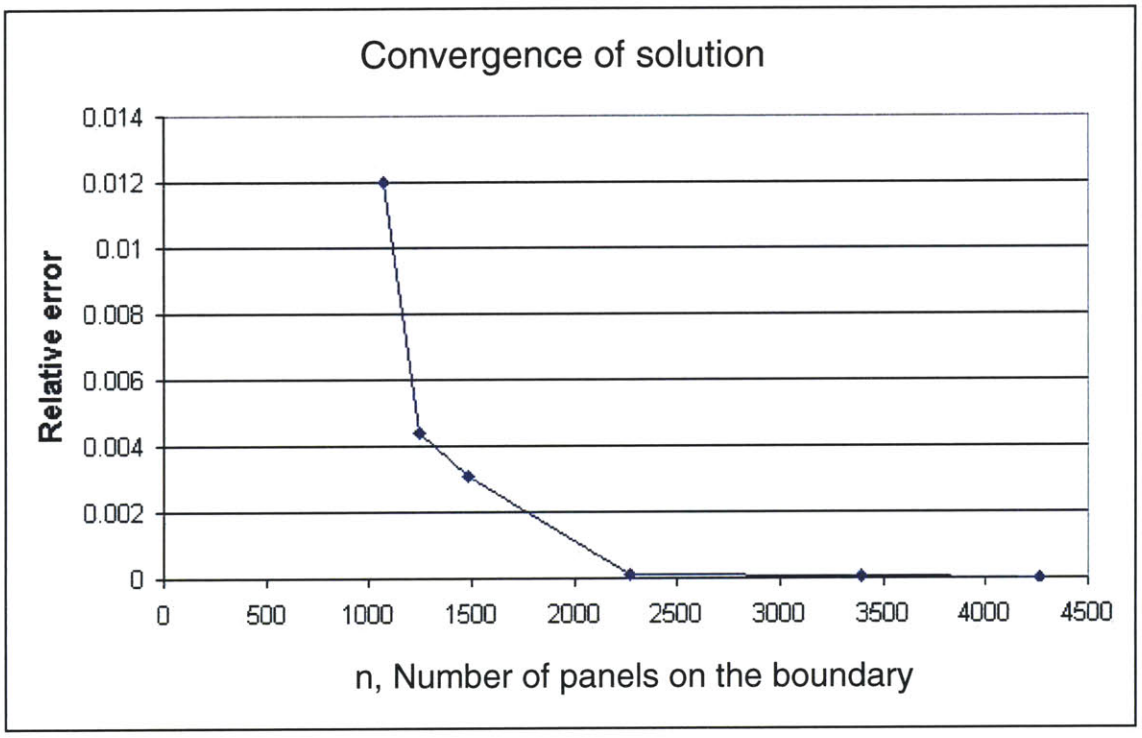


Figure 5-11: Convergence on example "Sample_split"

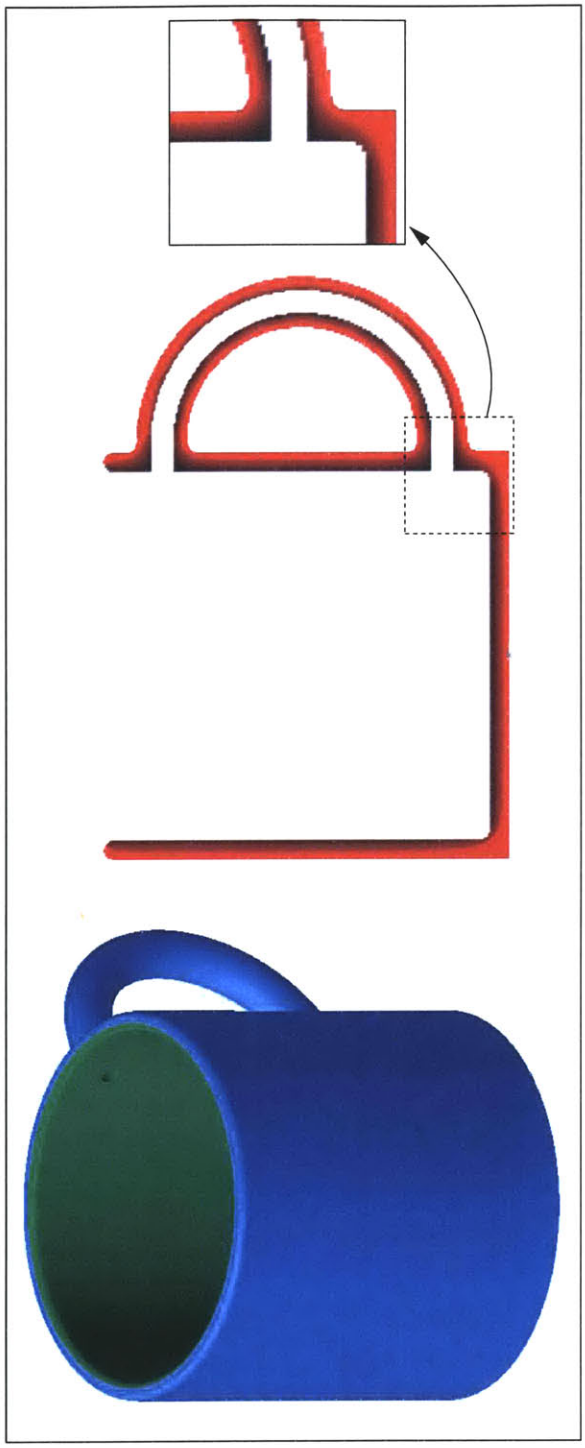


Figure 5-12: The "Mug" example with composition based on Laplace equation-based blending between its outer and inner surfaces

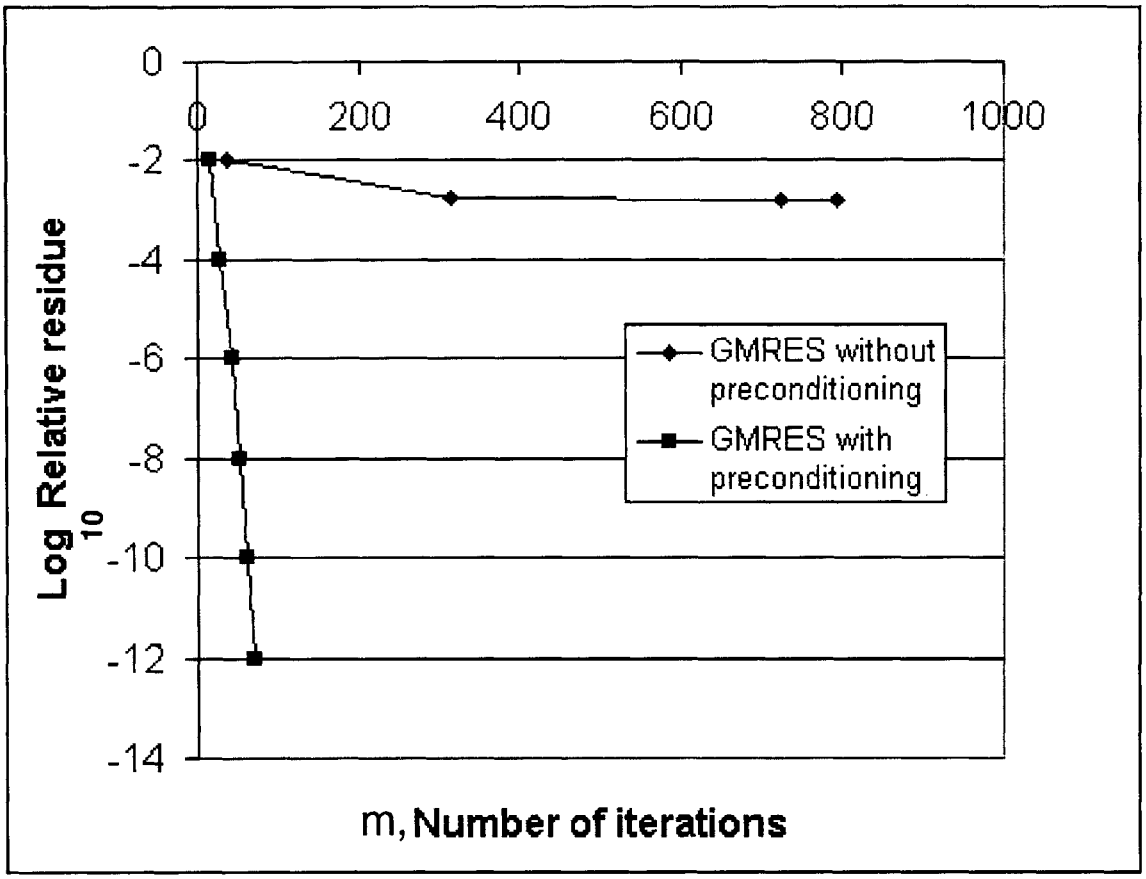


Figure 5-13: Convergence of GMRES solver with and without preconditioning on example "Mug"

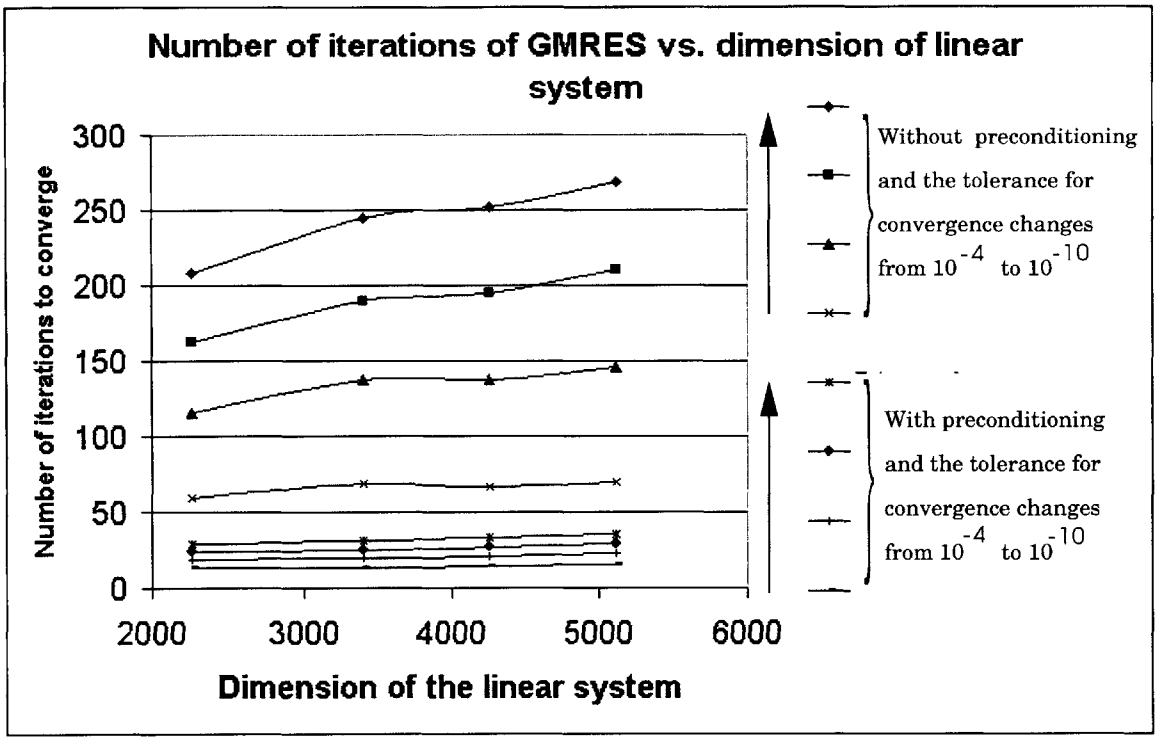


Figure 5-14: Number of iterations of GMRES solver with and without preconditioning on example "Sample_split"

Chapter 6

Feature-Based Model for LCC Solids

6.1 Introduction

In order to fulfill the third objective described in Section 3.1, this thesis develops a feature-based LCC object modeling approach based on an existing feature-based modeling system. As demonstrated in Figure 6-1, feature data in such a model are structured into five levels: a LCC assembly model, a LCC feature model, a part (component) model, a feature model and a generic model. The assembly model is the model at the highest level. Hatched arrows in Figure 6-1 represent the mapping of elements of a higher-level model to that of lower-level models.

In order to supply users with controls on the validity of a LCC solid, efficiency in updating the design after editing and efficiency in evaluation for visualization and outputs to drive the SFF fabrication, this thesis also develops a management model for the LCC object as illustrated in Figure 6-2. In Figure 6-2, SW refers to SolidWorks [74] that supplies geometric design events access to developers through its Application Programming Interface (API). In this manner, research resources are devoted to new tasks for which no commercial or prototype systems exist.

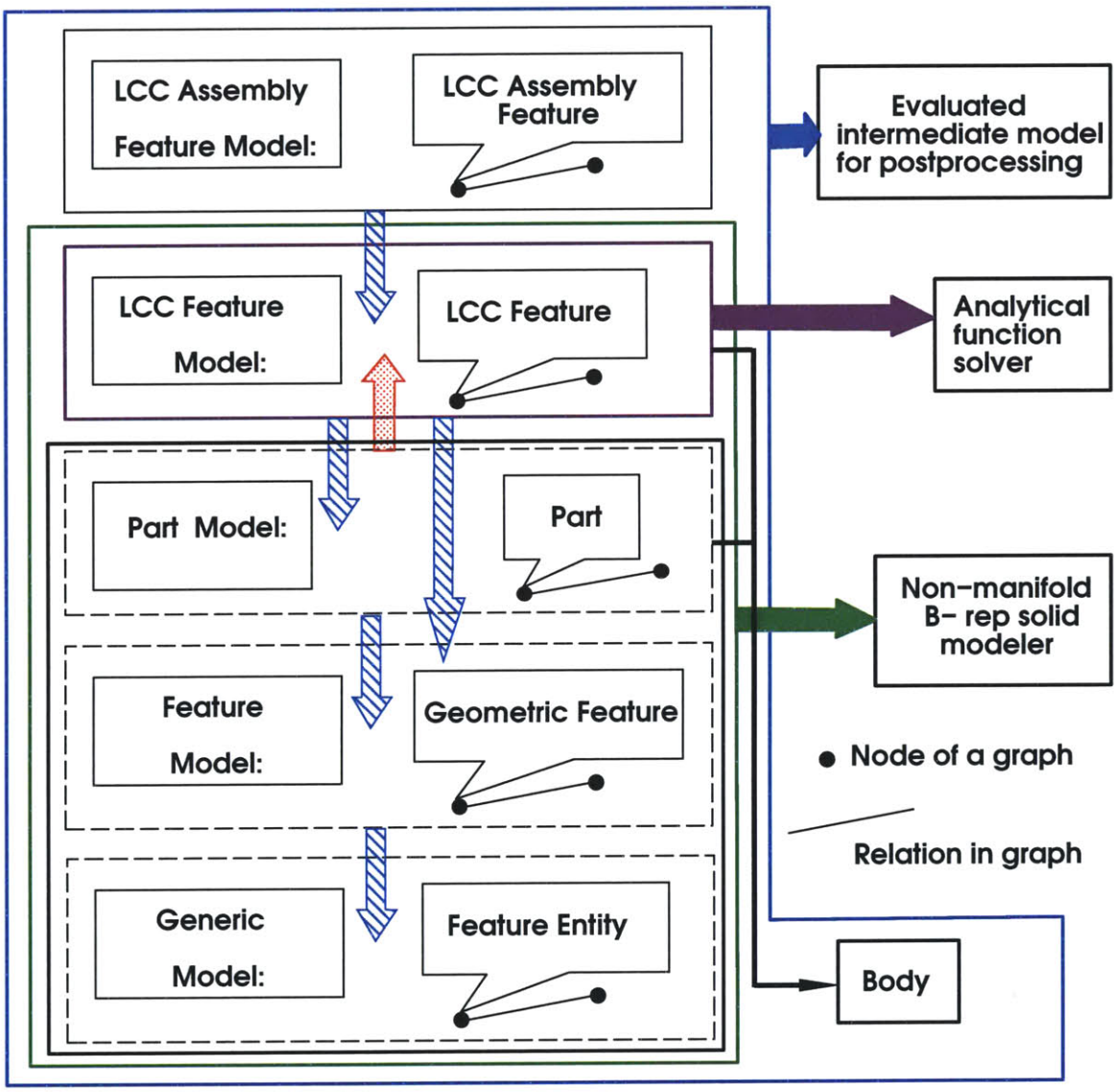


Figure 6-1: LCC feature model architecture

6.2 LCC feature model architecture

In the five-level model architecture, the bottom three levels are framed together with red lines because the representation of these levels are existing interior to the SolidWorks CAD system. The SolidWorks general graph representation is not visible or accessible from the feature manager. But the parent-child relation can be queried for a particular feature. The parent-child relationship in SolidWorks is defined as follows:

If a feature is based on some other features, then this feature is a child of the other features.

A feature can have as many children as well as many parents. Although some of the feature data and relations at the bottom three levels can be queried from SolidWorks API, one cannot reconstruct the graph representation because the API is designed to hide the real system's interior structure and it is not necessary to do that for the LCC design as long as the involved geometric feature objects are accessible either through the API (COM [20]) or through the user interface. In spite of this, it is necessary for a developer to understand such a feature representation. The nodes of the generic model are the topological or geometric entities or parameters that represent a feature. Topology constraints and geometric constraints are the linking relations in the graph, i.e. dimensional constraints [26].

In the feature model, features are mapped to sub-graphs of the generic entity graph and feature relations are mapped to a set of entity relations. Features contain the geometric, parametric or functional description of a feature. They can be typically classified into *geometric form features*, and *datum features* or *attribute features*. Section 4.2 can be referred to for various types of form features. Feature relations include relative positions, orientations, parametric dependencies between features, etc.

Parts are mapped to sub-graphs of the feature graph and part relations are mapped to a set of feature relations. Parts are also mapped to a body data structure (see Figure 6-3) via a Boundary Representation (B-Rep) solid modeling kernel, i.e. Parasolid [55]. The mapping is the procedure of derivation of the part shape from features. A part relation usually specifies a geometric or a non-geometric relation between two parts in an assembly. In the SolidWorks implementation, this kind of relation is superposed on component objects

in the assembly, while each body of a component is a linear transform of its referenced part. SolidWorks implemented two special part relations. One such relation is part derived from another part through external references, and the other (illustrated with red dotted arrow in Figure 6-1) is part referencing a specific component, within a specific assembly. These two external referencing mechanisms are very important capabilities of the system because in this fashion, part relations and parts can be created and edited in the context of an assembly.

The top two levels of the model are the local composition control (LCC) specific models. At the highest level is the LCC assembly model. LCC assembly is a graph with nodes representing various types of LCC assembly features and edges representing dependency relations between the LCC assembly features. The model also contains a queue of the LCC assembly features, which is following the assembly order in SolidWorks. The LCC assembly features identified in this thesis include some of the SolidWorks assembly features. However, these features are not enough for the purpose of LCC design. For example, the MateGroup feature in an assembly has a set of sub-features (mates), however if one component is a parent of a MateGroup, the relation does not transfer to the sub-features of this Mategroup feature. Actually assembly relations and external reference relations are not explicitly represented in this graph. The details about the graph are presented in Section 6.2.1.

The definition of a LCC feature has been illustrated in Figure 4-1 in Chapter 4. In specific terms, the geometric sub-feature in the definition of a LCC feature is a part (component) model in a feature based CAD system, which is SolidWorks for the system developed in this thesis.

The system is designed that a LCC feature also maps directly to features in the feature model because composition profile sub-features of the LCC feature are generally parametrized with respect to specific feature(s) in the model. Those specific features need to be referenced by the LCC feature. Such reference features could be features of the component of the LCC feature, or features that belong to components of other LCC features.

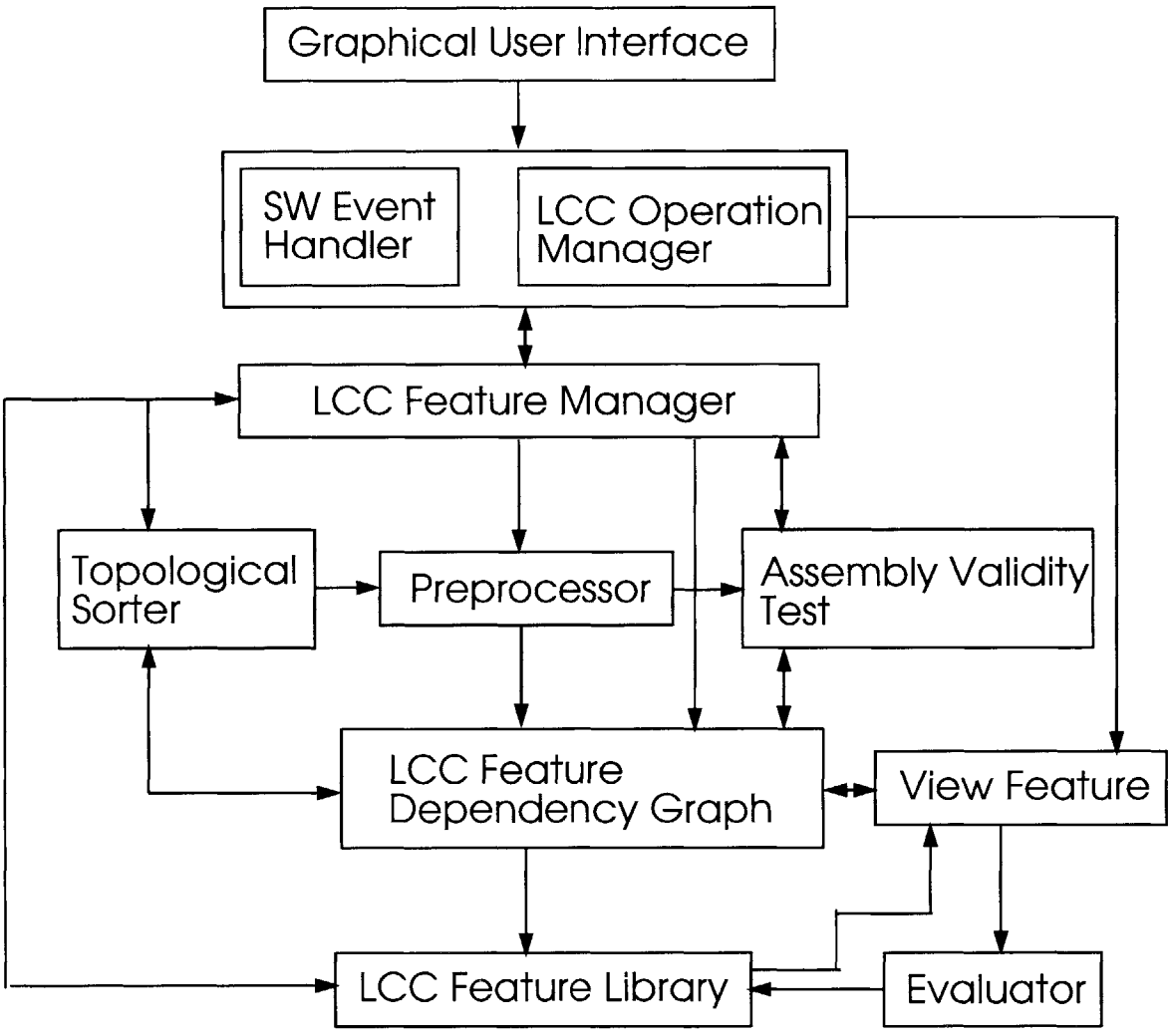


Figure 6-2: LCC feature model manager

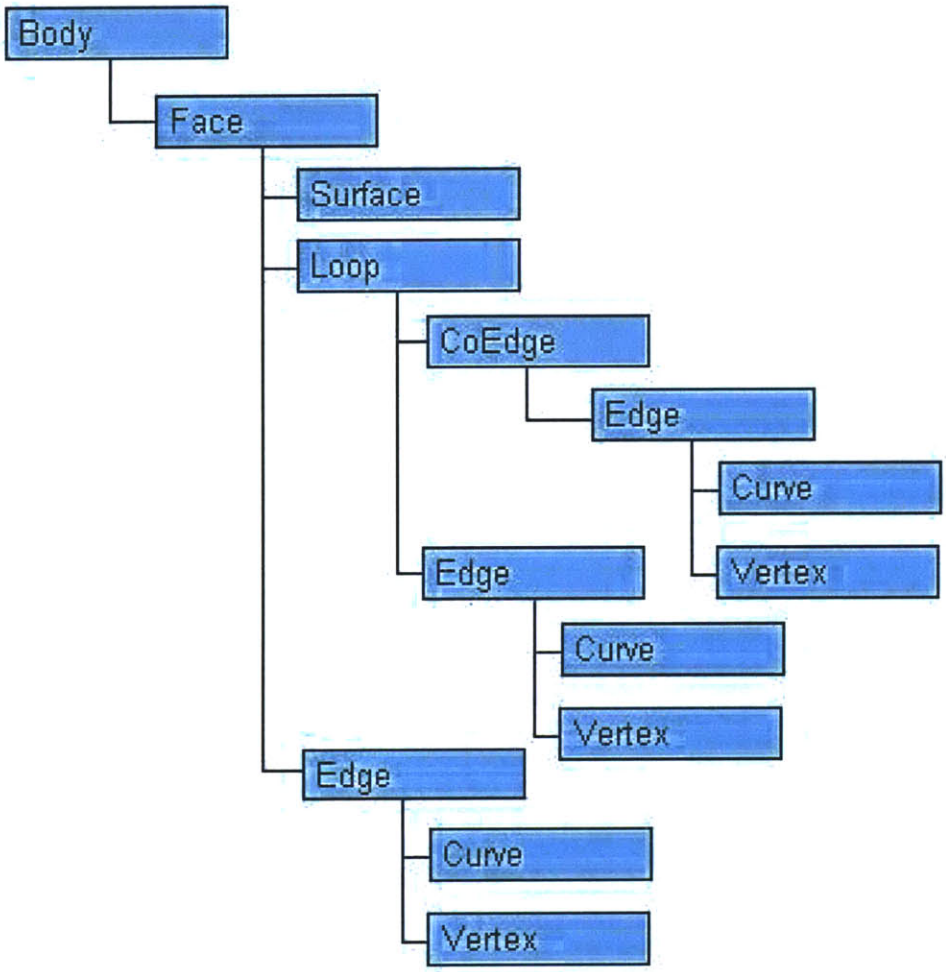


Figure 6-3: Topology data structure for a Body (adapted from [74])

6.2.1 LCC assembly feature dependency graph and LCC assembly feature interaction relations

The literature reviewed in Chapter 2, (for example [70, 28, 6, 29]) uses a graph-based model for a feature-based CAD system. This thesis also adopts a graph-based approach, because such an approach offers more efficiency or convenience in handling some modeling issues, such as:

1) Assembly model validity:

1. No isolated LCC feature — all the LCC features are assembled with features.
2. No degree of freedom — the root component is fixed, the rest are fully constrained.

2) Topological validity ([6]):

1. No closure (except for cavity, split); Closure causes some subtractive feature volume(s) to become a closed void inside the model.
2. No volume clearance; Volume clearance causes partial obstruction of the volume of a subtractive feature.
3. No boundary clearance; Boundary clearance causes (partial) obstruction of a closure face of a subtractive feature.

Examples of subtractive features include hole, cut, shell, etc.

3) Efficiently checking for material discontinuity.

4) Sequence in preprocessing the feature model (before visualization and postprocessing): Sorting in terms of material dependencies.

5) Editing induced break of dependencies and efficient updates.

6) Efficient evaluation of a LCC feature on a planar section or on a 3D lattice.

LCC assembly feature dependency graph is the top level of the LCC object model. This graph handles well issues 1, 4, 5 and 6 in the above list. It is a directed (connection relation

is unidirectional) graph [21]. For issues 2, 3 and 6 in the above list, an adjacency non-directed graph at the LCC feature level would be helpful. In this thesis, this part was not implemented. The nodes of the LCC assembly feature dependency graph are incidences of the types of single LCC-feature, LCC-pattern, LCC-fillet, Mate, in-context assembly (i.e. cavity), the geometric features that are referenced by LCC features and external reference features. This dissertation denotes these seven types of features with symbols L_1 to L_7. Among these, Mate and in-context assembly and external feature are rewrapped SolidWorks objects. The edges are the dependencies between them (parent-child relationship).

The following dependency relations are identified in this work:

(1) L_1 (single LCC component) $\rightarrow$ L_1 (single LCC surface) : the composition of the surface is calculated from the composition function of the component. The surface should intersect the volume of the component.

(2) L_1 (single LCC surface) $\rightarrow$ L_1 (single LCC component) : the composition of the surface defines the boundary condition of the composition of the component. The surface should be completely coincident with a subset of the boundary surfaces of the component.

(3) L_1 (single LCC component) $\rightarrow$ L_2 : the geometry and composition of the end node feature is patterned from the start node feature.

(4) L_1 (single LCC surface) $\rightarrow$ L_3 : the composition of the surface defines the boundary condition of the composition of the fillet component. The surface should be partly coincident with a subset of the boundary surfaces of the component.

(5) L_2 $\rightarrow$ L_1 (single LCC feature) : the pattern feature includes the component.

(6) L_6 (geometric feature used as reference) $\rightarrow$ L_1 (single LCC feature) : the geometric feature is used as reference for the distance-based composition design of a single LCC feature.

(7) L_1 (single LCC component) $\rightarrow$ L_4 : the component is assembled with a mate at some entity (entities) on the component.

(8) L_1 (single LCC component) $\rightarrow$ L_5 : the component is used in generating a geometric feature in relation to another component in the assembly.

(9) $L_3 \rightarrow L_4$: the fillet component is assembled with a mate at some entity (entities) on the component.

(10) $L_3 \rightarrow L_5$: the fillet component is used in generating geometric feature in relation to other components in the assembly.

(11) $L_7 \rightarrow L_1$ (single LCC component) : the external reference that consists of Part files that reference other Part files. An example of this type of reference would be a derived part from split.

(12) $L_7 \rightarrow L_3$: the fillet component is derived from an external reference.

These dependency relationships can be classified into composition related types and non-composition related types. Relations in (1)-(6) are composition related and (7)-(12) are non-composition related. Relations in (1), (2), (4), and (6) are not related to geometry and the rest are related to geometry.

In the following, the reason for the choice of the twelve dependency relations is described. The first two of the twelve are composition-related dependencies within the classes of single LCC-features. This thesis has not identified a meaningful dependency relationship between two incidences of LCC-fillet. All the rest five types of features are wrapped assembly features of SolidWorks, for which a dependency such as $L_n \rightarrow L_n$ neither exists in SolidWorks nor has meaning in the LCC application context. The number of choices for any combination of two types out of the seven types of features is C_7^2 which is equal to 21. Apart from the dependency relations that are identified, the following pairs of features are not related to each other: (L_2, L_3), (L_2, L_4), (L_2, L_5), (L_2, L_6), (L_2, L_7), (L_3, L_5), (L_3, L_6), (L_4, L_5), (L_4, L_6), (L_4, L_7), (L_5, L_6), (L_5, L_7), (L_6, L_7). Among the non-related pairs, (L_4, L_5), (L_4, L_7) and (L_5, L_7) are not indentified because mate, in-context assembly and external feature are rewrapped SolidWorks features and they do not have relations in SolidWorks and no specific composition feature can be attached to an incidence of one of these three types. The system was designed so that no filleting of a pattern and no patterning of a fillet

are allowed, therefore (L_2, L_3) is not identified as having a dependency relation. Patterns are not related to mate and in-context assembly feature in SolidWorks, therefore (L_2, L_4) and (L_2, L_5) are excluded from the indentified list as well. (L_2, L_7) is excluded because the system chooses to allow only local pattern in the assembly for the LCC application. L_6 is only specified for a composition feature as a reference, therefore it is not related to any feature that does not have composition feature attached, then the pairs (L_2, L_6), (L_4, L_6), (L_5, L_6) and (L_6, L_7) are excluded from the list. (L_3, L_6) is excluded because LCC-fillet is dependent directly on adjacent LCC-surface, but not on the selected geometric features. (L_3, L_5) is excluded because the system does not allow a fillet to be used for in-context geometry generation.

6.2.2 LCC model data structure

The data structure of the LCC model based on the architecture described in Figure 6-1 is presented here. Mostly it is the data structure for the top two levels of the model because the lower three are SolidWorks data that are accessed through its API. This thesis names an object that is modeled after the described architecture a LCC object. As demonstrated in Figure 6-4-(a), a LCC object in terms of the data structure maintains a graph that stores the data of the LCC assembly feature graph; a grid system and a slicer that are the intermediate models for the purpose of evaluation and visualization; and a queue of SolidWorks events or local composition control design operations. For each of the components of a LCC object except the event queue, further details are illustrated in Figure 6-4-(b)-(d).

The component 'TheDpdncyGraph' contains a stack of LCC assembly features, an adjacency matrix that represents the graph, a list to store the result for the topological sorting of the graph and the data for the types of edges of the graph. The component 'The Slicer' maintains a 2D map for composition values on a parameterized plane at certain resolution, therefore it contains also the parameters for the plane and slicing directions. The component 'The Grid' has stored a 3D map 'DT_map' for digital distance transform, a dither slicer 'The DitherSlicer' which contains a 2D image of the 3D grid, and a three-dimensional matrix as a buffer 'val_buf' for composition values at the grid points. The component 'EventOpsQueue' is a queue of event/operation info. Each 'EventOpsInfo' contains the

event identifying name (the same name of the matching LCC assembly feature in the graph later), related component's name, and the type of the event/operation. In this thesis, five types of event/operation are defined types such as 'event_ops_add', 'event_ops_delete', 'event_ops_edit_geo', 'event_ops_edit_cmp', and 'event_ops_invalidated'. The first type represents any event that adds an incidence of LCC assembly features, which are defined in Section 6.2.1. The 'event_ops_delete' represents the type of event or operation that deletes an incidence of any LCC assembly features. The 'event_ops_edit_geo' denotes the type of events that edit a LCC assembly feature geometrically. The 'event_ops_edit_cmp' denotes the type of operations that edit a LCC assembly feature in terms of material composition feature. The 'event_ops_invalidated' represents the type of events or operations that invalidate a LCC assembly feature.

As can be seen, there exist different types of LCC assembly features, LCC features and LCC Composition features identified in this thesis. These classes of features form feature libraries which are demonstrated in Figure 6-5 to Figure 6-7. The library of the 'LCC_Assem_Feature' relates to the Library of the 'LCC_Feature' through mapping. The generic 'LCC_Assem_Feature' class contains a pointer to 'LCC_Feature' as a member, therefore in this way it is mapped to a single LCC_feature. The type 'LCC_Pattern' is mapped to 'LCC_feature' through the seed component object and the patterned component objects. 'LCC_Mate' is mapped to 'LCC_feature' through the SolidWorks 'Mate' feature, which contains the information of the associated components. Similarly, 'LCC_InContextFtr' is mapped to 'LCC_feature' through the SolidWorks 'InContext Feature', and the 'LCC_ExternalReference' is mapped to 'LCC_feature' through the SolidWorks 'External Reference feature'. 'LCC_Assem_Ref_Feature' is mapped to 'LCC_Ref_Face' through the SolidWorks geometric feature it contains. It should be noted that a special single 'LCC_feature', 'LCC_fillet', is mapped from 'LCC_Assem_Feature' through a SolidWorks component and several 'LCC_Ref_Face's.

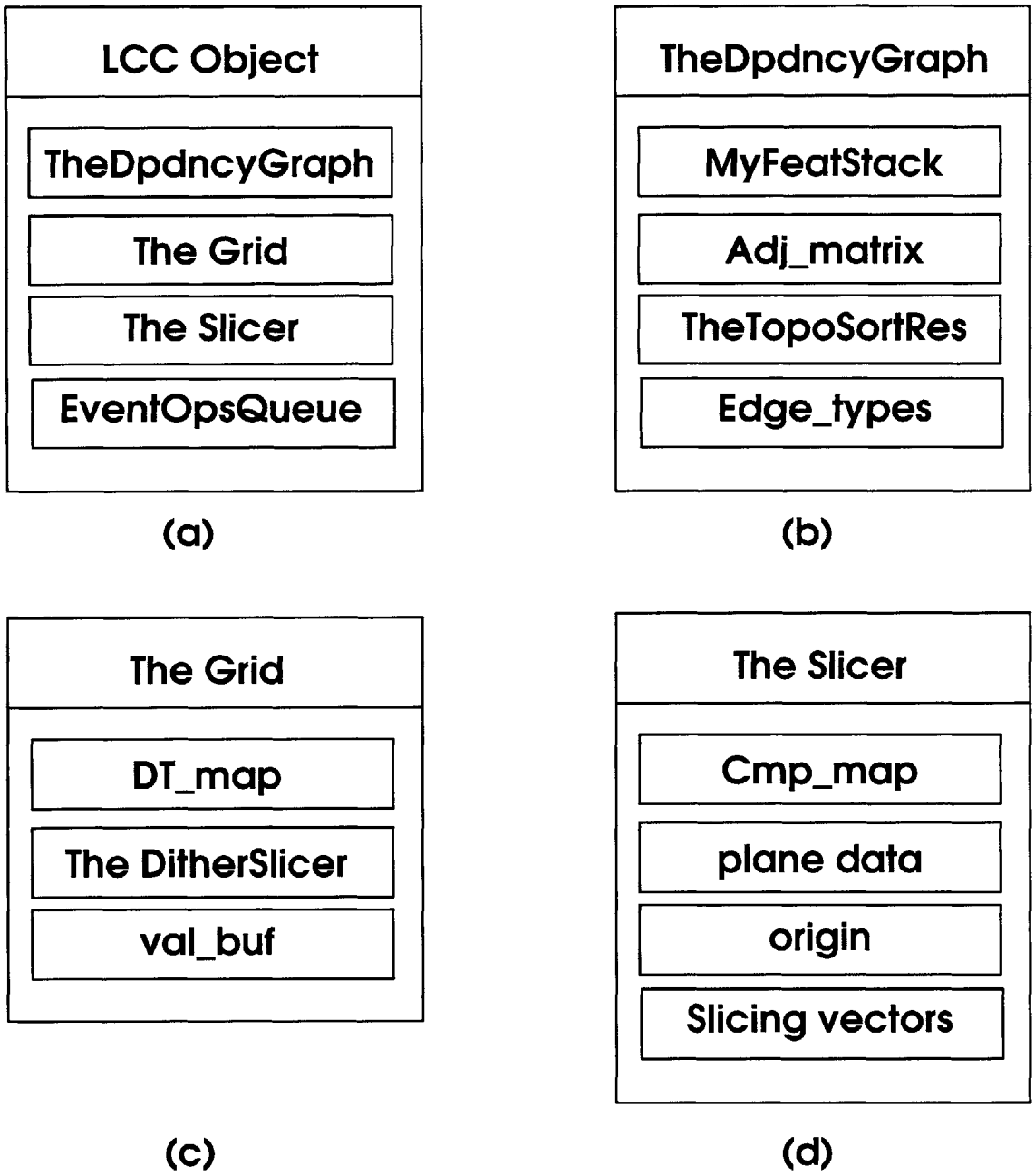


Figure 6-4: Data structure of the LCC model

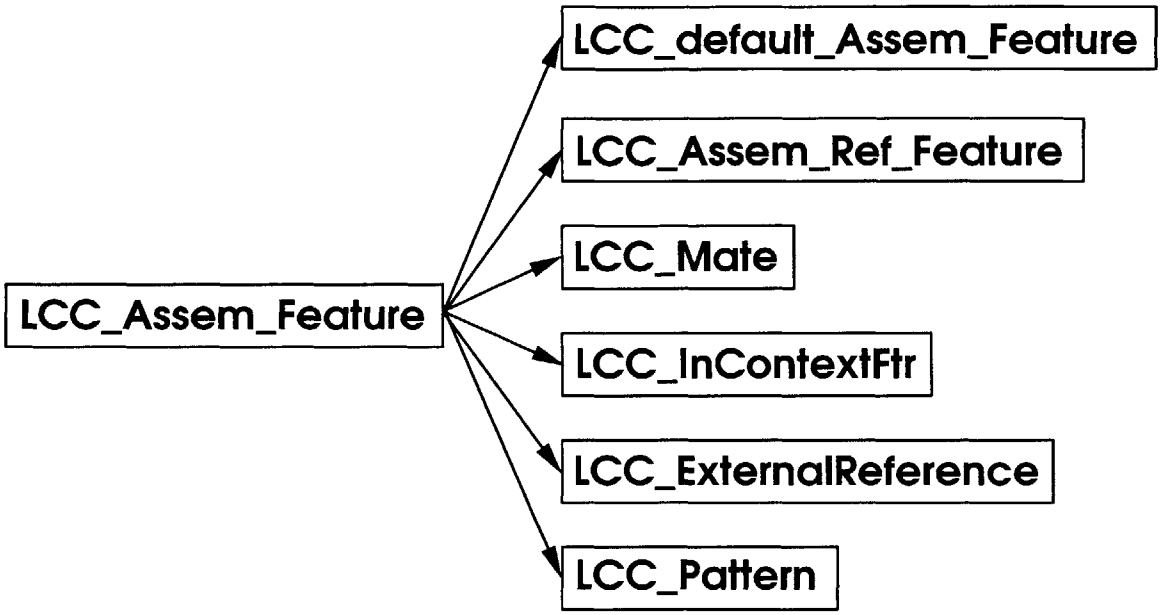


Figure 6-5: Library of the 'LCC_Assem_Feature'

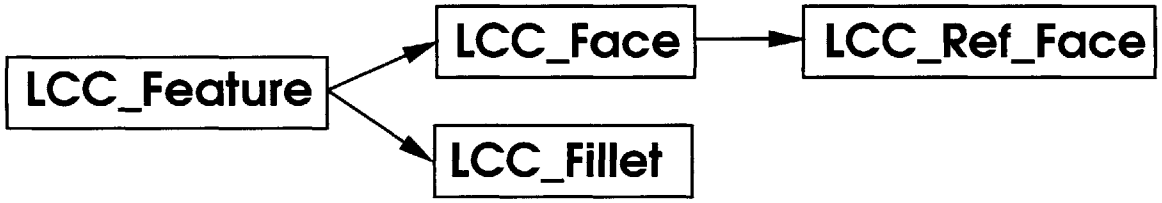


Figure 6-6: Library of the 'LCC_Feature'

6.3 LCC model manager and algorithms for maintaining LCC model

6.3.1 LCC model manager

In this section, the framework illustrated in Figure 6-2 for managing the graph based LCC object representation is presented. By managing, we refer to all creation, editing, preprocessing, validity checking operations. In Figure 6-2, the arrows represent the information flow in the process. In the following, the block components in the diagram for the LCC model manager are described.

SW Event Handler and LCC Operation Manager: This handles the Graphical User Interface (GUI) feature events in the environment of the SolidWorks system and translates

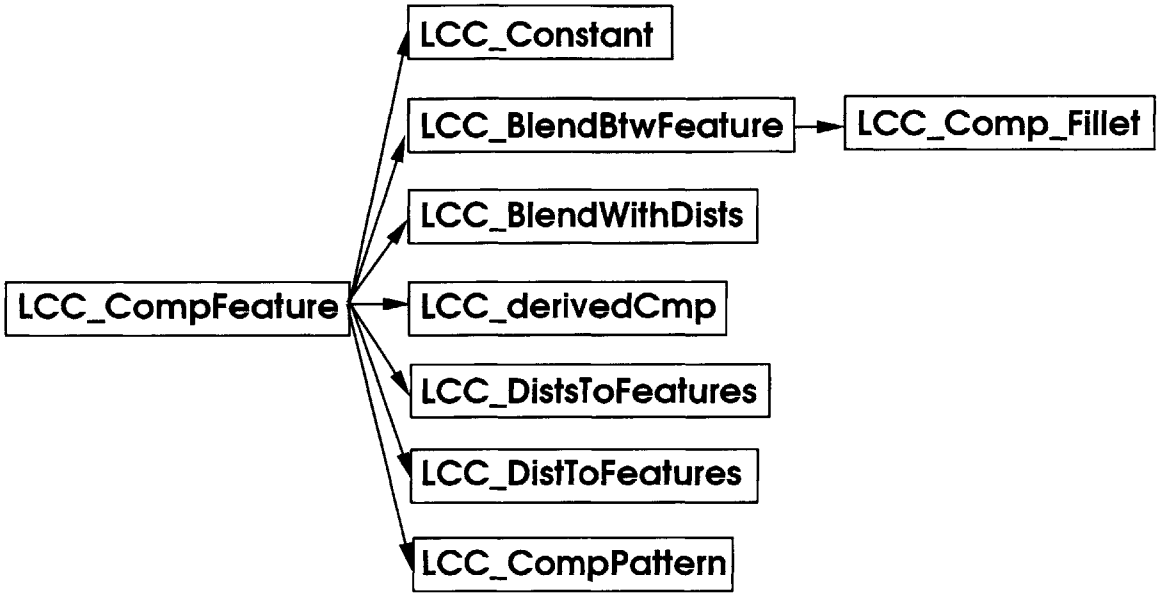


Figure 6-7: Library of the 'LCC_Comp_Feature'

the LCC Operations on SolidWorks feature and components into addition of LCC feature in the LCC feature dependency graph, editing of LCC feature or deletion of LCC feature. It also handles the LCC view events to connect to View Features. This component also handles the adding of different types of LCC assembly features.

LCC Feature Manager: Check whether the edited feature is nominally valid, so that only validly defined features are edited; Manage the graph - i.e. construct or update it given each addition of a feature at a node, editing on the feature at a node or deletion of the feature at a node. After each operation or event, the graph is checked for the following:

(a) the graph is preprocessed with a topological sorter [21], and then the individual features are preprocessed according to the sequence of the sorting result when necessary. It also checks if the involved LCC features can be preprocessed successfully.

(b) check whether each individual feature is nominally valid.

(c) check whether the LCC assembly model is valid.

In the case of a distance-based design, bucket sorting and digital distance transform algorithms are the preprocessing and in the case of a Laplace equation based blending design, BEM method is the individual preprocessing.

Topological sorter: Sorting the feature dependency graph to a linear order of all its vertices such that if v is dependent on u, then u appears before v in the ordering. This helps in efficient update of features, efficient preprocessing, efficient evaluation, (ie. efficient ray-casting of the assembly after editing).

Preprocessor: The module that is called by LCC feature manager to preprocess the individual LCC features in the graph, if a preprocessing algorithm such as distance transform, or BEM are required. The preprocessing is only done on the feature that needs it in the interactive procedures and only according to the sequence of the sorting result. This module helps to check the validity of features after preprocessing.

Assembly validity tester: Test if the assembly model meets the criteria that there is no isolated LCC feature and there is no degree of freedom, and test if individual features are valid during the process. See also Section 6.2 for a detailed description.

LCC Feature Library: The complete set of classes of LCC assembly feature, LCC features and LCC composition features that are developed through inheritance.

View feature: It includes user input parameters for the view and data for storing the view and the relations between view data and individual features in the graph, and the methods for the interaction between the graph and this view.

Evaluator: It takes care of the evaluation of the view feature; the details are given in Chapter 7.

6.3.2 Algorithms for maintaining the LCC model

The algorithms for maintaining the LCC model are presented in the following. The flow of the design procedures are described, then the scheme of operations is described and the algorithms for the manager are presented. Reference [48] is a guide for the developed system which also helps in the understanding of the algorithms. SolidWorks [74] was chosen for the

system development because it is a state-of-art feature-based CAD system and also because it provides an API tool that is rich enough to accomplish the objective of this thesis and research resources can be saved in this way.

Flow of the design procedures

The flow of design procedures is as follows:

1. Design of geometry
2. Material system design
3. Local composition control feature design
4. Viewing LCC object to verify results

Design of geometry

Feature-based design of geometry is the process of creating a user-intended geometric part with a series of features that are provided in a feature-based parametric design system such as SolidWorks [74]. The features in such a system can be categorized into the following three: generated features, modifying features and datum features [28]. *Generated features* such as *extrusion, revolution, sweeping,* etc. usually are generated from user-defined sketches. *Modifying features* includes *fillet, chamfer,* etc. *Datum features* are features used as references.

Specifically, this LCC design tool is an addIn to SolidWorks, therefore the user can utilize SolidWorks features and functionalities (see SolidWorks documentation [74]) to generate the intended geometric part. The part document in SolidWorks will maintain a tree of features for the design. The user of the LCC design tool is supposed to insert the parts into an assembly document and assemble the components with assembly features. Components can also be created by splitting one component with several parametric surface features. Components can also be edited in the environment of assembly. Components can also be newly created or edited in the context of an assembly through an external reference. During the session of assembly design in SolidWorks, both the features in the assembly and the

features in the part document will be maintained in the feature tree. Therefore requiring the user to work only on the assembly document does not limit design capabilities.

Material system design

The material system design is the procedure of specifying an array of materials each possessing some properties. In this prototype system, the material properties are names and RGB colors associated with each material. The system allows the user to specify a default RGB color for each material so that when no material features are assigned to a geometric component, the default constant values apply. In comparison with SolidWorks color options, the material colors apply to volumetric components instead of surface features. The user can adjust the color values for each material during the design process.

Local composition control feature design

The user is allowed to select a volume which is in the form of a SolidWorks component, or a surface feature, or a component pattern, or a special volume (volume fillet), which is one of the components split off a part with user defined surface features and apply composition features onto them. According to the type of geometric domain the composition features are applied, local composition control features are classified into: LCC volume, LCC surface, LCC pattern, and LCC fillet. With the variety of classes of features presented in Chapter 5, and the large number of degrees of freedom available through composition function profile specification, a wide range of applications can be created via LCC features. It should be noted that for a desired LCC object, there may not be a unique combination of features that can be used. It is up to the user to design the LCC object with LCC features in the most efficient way.

Viewing LCC object to verify the LCC design

After the LCC composition design, it is necessary to verify the design with visualization. This design tool allows the user to visualize the composition of the LCC object on a user-defined plane, the composition of the LCC object on the outer boundary of the object and the composition on layers of a 3D grid defined over the object with user input resolutions. The underlying algorithm is based on the ray casting of the LCC object.

The scheme of operations

The scheme of the model manager's response to a SolidWorks event or a LCC operation is demonstrated in Figure 6-8. The manager's event and operation handler checks the nominal validity for geometric feature or composition feature, then the LCC feature manager checks the nominal validity for LCC features that are designed through the events and operations. Then the LCC feature manager updates the graph with LCC features created or edited by the operation. Afterwards, the topological sorter sorts the graph and generates a sequence of LCC features for preprocessing and also checks if the graph has unallowable cycles. The preprocessor of the manager preprocesses the different LCC features as necessary. Last, the validity tester checks the validity of individual LCC features and of the assembly.

Algorithms

The algorithms presented here are primarily the algorithms dealing with the third phase —"Feature-based local composition control feature design" in the design flow introduced in Section 6.3.2.

One time processor of a SolidWorks assembly document: Once an assembly document, the user chooses to work on, is opened or newly created, the user is supposed to design the material system with RGB colors as parameters for each material. In order to do the local composition control design, the user is to select the initialization of the LCC design, which does a one time processing of the SolidWorks assembly document so that an LCC feature-based model for the solid is created. Figure 6-9 demonstrates the relations that are connected when a LCC pattern is added. Algorithm 6 is the algorithm for this processor.

Algorithms for handling the SolidWorks events in the design: Event notifications invoked by different SolidWorks operations are listed below:

Add a single component — notification(s): OnAddItem(...).

Add a pattern — notification(s): OnRegen(), OnAddItem(...)(s), OnPostRegen().

Add a mate — notification(s): OnRegen(), OnPostRegen().

Algorithm 6 Initial processing of SolidWorks (SW) assembly document

get the root component of the assembly document;
recursively access each component and add a corresponding single LCC-feature (a type of LCC assembly feature) with default composition values into the LCC assembly feature graph.
if a component has external reference **then**
  get the component in the external reference list;
  if check if an LCC external reference with this name has existed **then**
6: find the LCC assembly feature that corresponds to the component and direct the edge from the LCC external reference to it in the graph;
  else
    add a new LCC external reference with the SW external reference information;
    find the LCC assembly feature that corresponds to the component and direct the edge from the LCC external reference to it in the graph;
get the first feature in the SolidWorks feature tree for traversing;
for each feature in the tree **do**
12: **if** the feature is a SolidWorks mate **then**
    get the two components that are related by this mate;
    get the two corresponding LCC assembly features in the graph;
    add the LCC mate into the graph;
    direct the edge from those two LCC assembly features to the mate feature in the graph;
  if the feature is a SolidWorks pattern (Figure 6-9) **then**
18: add the corresponding LCC pattern feature into the graph;
    find the existing assembly feature in the graph that contains the seed component of the pattern;
    connect the edge from the assembly feature that contains the seed to the LCC pattern feature;
    get the components of the pattern and find their corresponding assembly features in the graph;
    set the main feature of the assembly features with component LCC features of the pattern;
    connect the edges from the LCC pattern to those assembly features in the graph;
24: **if** the feature is a SolidWorks inContext feature **then**
    get the SolidWorks external reference list for this inContext feature;
    get the component in the external reference list;
    get the related real feature in the component, i.e. the Cavity;
    add the LCC inContextFtr into the graph;
    get the base component of cavity feature, connect its assembly feature to the LCC inContext feature in the graph;
30: get the referenced entities, i.e. the components in the pattern
    check if the entities are components, find the assembly features corresponding to these reference entities;
    connect the edges from them to the LCC inContext feature in the graph;
Count the number for each type of features.

Add in-context assembly feature (i.e. the one for "cavity") — notification(s): OnBgnInContext(...), OnRegen(), OnPostRegen(), OnEndInContext(...).

Add geometric feature referenced by LCC features — if it is an assembly level feature, no notification is sent by SolidWorks; if it is part level feature: if a sketch is added, during the editing of component, OnRegen(), OnPostRegen() are called; if a feature (i.e. Boss, Fillet) is added, no notification is sent, but SolidWorks prompts that rebuilding is necessary and if the user chooses to rebuild, then OnRegen(), OnPostRegen() are issued.

Add external reference — no notifications are issued, but the event can be detected in the process of addition of component, redundancy need to be deleted.

Edit a component in the context of an assembly — notification(s): OnBgnInContext(...), OnRegen(), OnPostRegen(), OnEndInContext(...).

Edit a pattern — notification(s): OnPreFeatureEdit(...), OnRegen(), OnPostRegen().

Edit a mate — notification(s): OnPreFeatureEdit(...), OnRegen(), OnPostRegen().

Edit a feature at the assembly level (i.e. a reference surface) — notification(s): OnPreFeatureEdit(...).

Edit a feature of a component — notification(s): OnPreFeatureEdit(...), OnBgnInContext(...), OnRegen(), OnPostRegen(), OnEndInContext(...).

Delete a component — notification(s): OnDeleteItem(), OnRegen(), OnPostRegen().

Delete a pattern — notification(s): OnDeleteItem()(s), OnRegen(), OnPostRegen().

Delete a mate — notification(s): OnRegen(), OnPostRegen().

Delete a feature at the assembly level — notification(s): OnRegen(), OnPostRegen().

Delete a feature of a component — notification(s): OnRegen(), OnPostRegen().

The meanings of the notifications are as follows:

OnAddItem(...) This event is issued when an item is added to one of the SolidWorks "manager trees." The item is either a component or a configuration.

OnBgnInContext(...) This event notifies the application that the user is starting to edit an assembly component within the context of the assembly (inside the assembly document window).

OnEndInContext(...) This event notifies the application that the user is starting to edit an assembly component within the context of the assembly (inside the assembly document window).

OnDeleteItem(...) This event is issued when an item is deleted from one of the SolidWorks "manager trees".

OnRegen() This event pre-notifies the user program when an assembly document is about to be regenerated.

OnPostRegen() This event post-notifies the user program when an assembly document has been regenerated.

OnPreFeatureEdit(...) This event pre-notifies the user program when the user edits the definition of a selected feature.

OnRenameItem(...) This event is issued when an item is renamed in one of the SolidWorks "manager trees".

OnNewSelection() This event post-notifies the user program when the selection list has changed.

OnFeatureSketchEdit(...) This event pre-notifies the user program when the user edits the definition of a sketch.

Algorithm for handling the composition design operations: Figure 6-10 demonstrates the relations that are connected when a LCC fillet is created. Such relations are defined rules for a LCC fillet feature.

For the LCC Feature Manager

The reaction of the manager to a deletion of a LCC assembly feature is the following:

Algorithm 7 Algorithm handling SolidWorks events in the design.

for each SolidWorks event notification **do**
listen to find out which type of operation is happening to which type of features; ▷see Algorithm 8
test the validity of the feature (geometrically) at this stage. Do corresponding reaction to the SolidWorks data;
store the feature and the type of operation in the EventOpsInfo queue for the LCC feature manager to process further.

Algorithm 8 Algorithm for identifying which operation happens to which feature:

By the counter for features, operation can be detected to be adding, editing or deleting. It will be found out for different features, an adding operation will invoke what types of notifications. Use the results to identify the feature operations.
Similarly for editing and deletion operations, the above method applies.

STEP 1 Recursively deal with the children of children of the LCC assembly feature in the graph.

STEP 2 If no children to deal with any more and if the relation type is geometry-related, then delete the children.

STEP 3 If no children to deal with any more and the relation is purely composition-related, then set the composition of the children feature as default.

STEP 4 Delete the LCC assembly feature.

The reaction of the manager to geometric editing of a LCC assembly feature is the following:

STEP 1 Check out the warnings issued by SolidWorks and handle with the corresponding reaction for invalidated LCC assembly features.

STEP 2 For each geometrically related feature in the graph, check if the prescribed rules of feature relations are in order. If not, then react accordingly.

Editing of the geometry of a LCC assembly feature often does not violate the features' definition/validity. There are two cases where special handling is necessary. One scenario is the editing of one geometry feature causes some geometric feature in the model to become unsolvable and SolidWorks usually gives warning for this case. An example for this case is

Algorithm 9 Algorithm handling the composition design operations

if a LCC composition feature is designed (added) **then**
identify to which SolidWorks feature and what type of feature the composition feature is assigned, the feature has to be existing as a default LCC assembly feature in the graph (assigned during the batch process in algorithm 6.)
check if the composition feature is nominally valid;
if the operation is "Assign composition feature" **then**
if the selected geometric feature for composition design reference does not exist in the graph **then**
create an "LCC_Assem_Ref_Feature" feature with this geometric feature;
add the created feature into the graph;
direct the edge from the composition reference feature to the single LCC assembly feature
if the operation is "Add LCC Surface" **then**
if the to-be-added surface feature does not exist in the graph **then**
create an "LCC_Assem_Feature" feature with this surface feature;
add the created feature into the graph;
if the operation is "Add LCC RefSurface" **then**
if the to-be-added reference surface feature does not exist in the graph **then**
create an "LCC_Assem_Feature" feature with this reference surface feature;
add the created feature into the graph;
if the operation is "Add LCC Fillet" **then**
if the surface features for cutting do not exist in the graph **then**
create "LCC_Assem_Feature" features with these surface feature;
add the created features into the graph;
store the feature and the type of operation in the EventOpsInfo queue for the LCC feature manager to process further;
if a LCC composition feature is edited through the right mouse button at the feature **then**
check if the feature is editable compositionally;
check the nominal validity of the edited composition feature;
store the feature and the type of operation in the EventOpsInfo queue for the LCC feature manager to process further;
if a LCC composition feature is deleted through the right mouse button at the feature **then**
check if the composition feature can be deleted (patterned components and fillet's composition feature are not deletable);
delete the composition feature and replace with default composition;
store the feature and the type of operation in the EventOpsInfo queue for the LCC feature manager to process further.

Algorithm 10 Algorithm for managing the graph model

```
get the invalidated LCC assembly features;
topologically sort the graph;
for each event/operation in the TheEventOpsQueue do
  if theType == event_ops_delete then
    call the function of deleting a LCC_Assem_Feature from the graph;
  if theType == event_ops_edit_geo then
    call the function of editing a LCC_Assem_Feature geometrically;
  if theType == event_ops_edit_cmp then
    call the function of editing a LCC_Assem_Feature for composition;
10: if theType == event_ops_invalidated then
    prompt the user to take action;
    if the user chooses to delete then
      call the function of deleting a LCC_Assem_Feature from the graph;
if everything was handled well then
  reset the event/ops queue;
  topologically sort the graph;
  preprocessing in the order of the sorting result;
  if preprocessing of each LCC assembly feature is successful then
    check the assembly validity;
20: prompt the user about the status;
else
  keep the unhandled events in the queue and prompt the user to make changes.
```

when a pattern is changed by replacing the seed component with another component and the original patterned components were used to define a cavity. Then the cavity becomes unsolvable in SolidWorks after the editing. The other special case is when the editing of the geometry of the LCC assembly feature causes the violation of the rules of feature relationship specified for the LCC assembly features in this thesis. An example is when a pattern is changed by increasing the number of its patterned incidence and the original patterned components were used to define a cavity. Although the SolidWorks system finds no problems in this case, the LCC pattern needs to be updated in the graph in terms of the relationship that is illustrated in Figure 6-9.

The reaction of the manager to an editing of the composition feature of a LCC assembly feature is the following:

STEP 1 Recursively deal with the children of the LCC assembly feature, if the relation is composition-related.

STEP 2 If any geometric reference feature is added or deleted in the process, then call the corresponding reaction functions of the manager.

One example of induced deletion and addition of a geometric reference feature is when a distance-based composition profile is assigned to a different geometric feature, which is essentially a deletion of the old reference feature and an addition of a new reference feature.

For the topological sorter:

In the context of feature-based LCC modeling, the following two specific objectives need to be achieved:

1. No feature dependencies should be cyclic.

2. For the maintenance and evaluation of the feature model, precedence order should be determined.

In terms of the algorithm, the topological sorter has the tasks to check if the graph has an unallowable cycle and to determine a precedence order. The precedence order has been explained in section 6.3.1. A cycle is defined as a path that is simple except that the first and last vertex are the same. A path from vertex x to y in a graph is a list of vertices

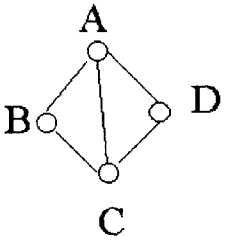
in which successive vertices are connected by edges in the graph and it is simple when no vertex is repeated.

Computer representation of a graph is usually in the form of an adjacency matrix or adjacency list. Figure 6-12 and the upper right of Figure 6-11 demonstrate these two types of representation, respectively.

The storage cost of the adjacency matrix is of $O(V^2)$, where V is the number of vertices of the graph. The storage cost of the adjacency list is of $O(V+E)$, where E is the number of edges of the graph. Since the number of features in the graph is not expected to be large, the adjacency matrix was chosen for the graph representation. Vertices are stored in a stack, each vertex points to an instance of one type of LCC features.

Topological sorting is a classic graph algorithm based on depth-first search (DFS) [21], which helps solve the above two problems. "Depth first search" searches "deeper" whenever possible for exploration of edges and creates a depth-first forest, where forest is defined as a set of disconnected trees and a tree is a graph without cycle. During the search, each vertex is attributed as undiscovered (white), discovered (gray), or finished (black) and also time stamped with discover time $d[v]$ and finish time $f[v]$. The upper left of Figure 6-11 demonstrates how the algorithm works and the numbers at each node are the discover and finish times.

The lower right of Figure 6-11 shows that graph edges can be classified into four types, i.e. tree edges, back edges, forward edges and cross edges. Tree edges are edges in the depth-first forest. Edge (u,v) is a tree edge, if v was first discovered by exploring edge (u,v). Back edges are edges (u,v) connecting a vertex u to an ancestor v in a depth-first tree. Self-loops are also back edges. Forward edges are non-tree edges (u,v) connecting a vertex u to a descendent v in a depth-first tree. Cross edges are all edges other than the above three kinds. The algorithm for classification of edges is as follows:



Adjacency matrix

	A	B	C	D
A	1	1	1	1
B	1	1	1	0
C	1	1	1	1
D	1	0	1	1

Figure 6-12: The adjacency matrix representing a graph

When the edge is first explored,

- If color is white, then edge is tree edge;
- If color is gray, then edge is back edge;
- If color is black, then edge is forward or cross edge.

Based on the Lemma that a directed graph G is acyclic if and only if a depth-first search of G yields no back edges [21], the problem "does the graph have cycle?" can be solved by checking if there is a back edge in the DFS trees. The precedence list can be determined through sorting vertices by their finish time in the DFS. Appendix A.3 gives a pseudo-code for the basic algorithm. The time complexity of this algorithm is of $O(V + E)$, where E is the number of edges of the graph.

In the following, the algorithm for checking if an unallowable cycle exists in the graph is presented. Once a back edge is identified in the depth-first search procedure, this means that a cycle exists in the graph. However, because there are different types of feature relation meanings for the edges of the LCC feature graph, a cycle is not necessarily invalid for the representation. Only the case where all the edges in a cycle are in the same category is an unallowable cycle. The three categories are pure composition type, pure geometric type and both composition and geometric type. The pseudo-code of the algorithm is shown in Algorithm 11. The method is a variant of a depth-first search algorithm after the first round depth-first search that has identified the back edges and edge categories.

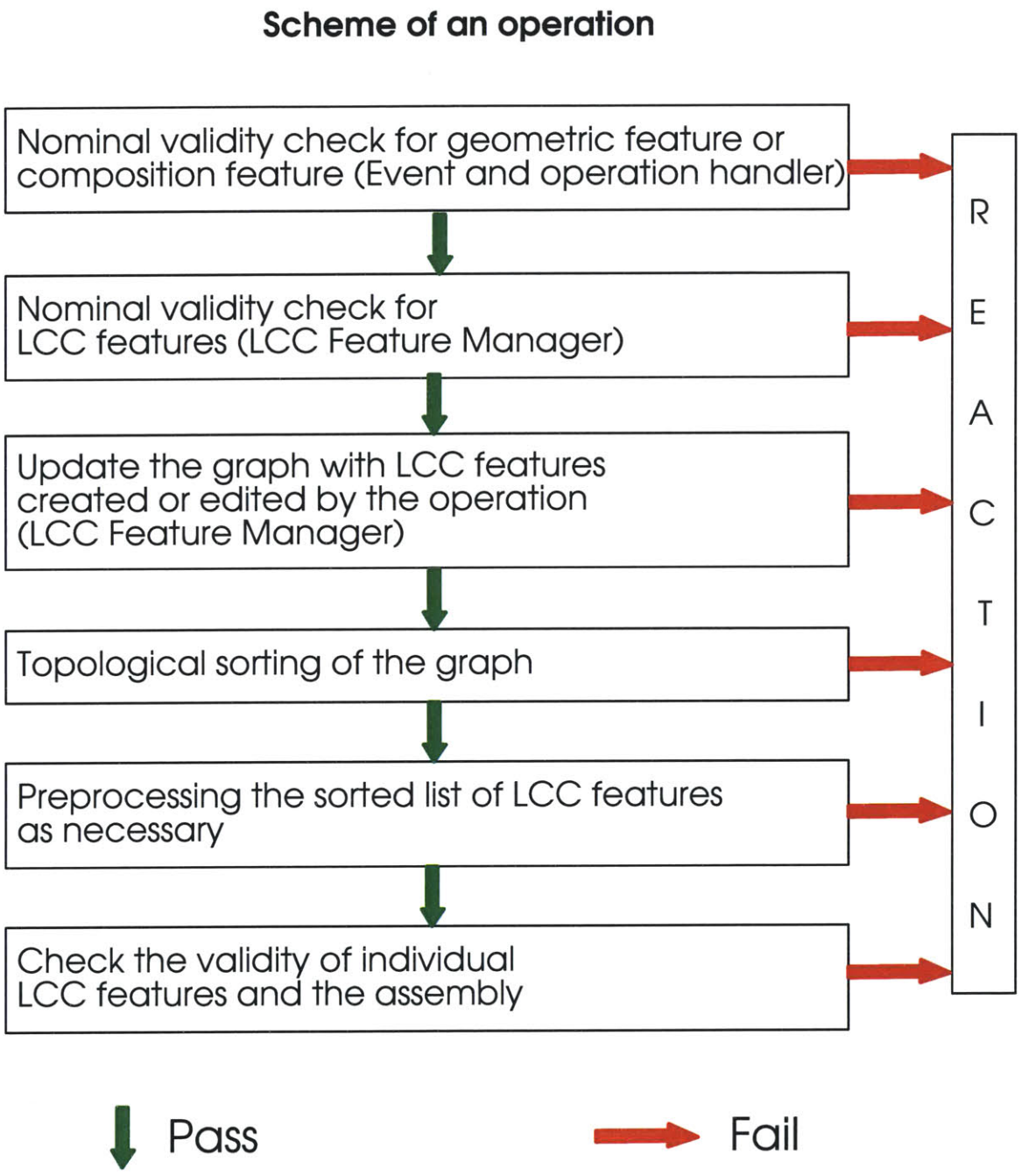


Figure 6-8: Scheme of an operation

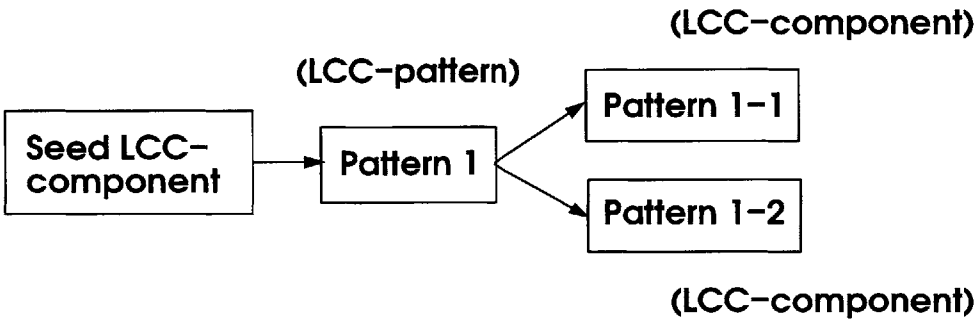


Figure 6-9: The subset of the graph that relates to a LCC pattern feature

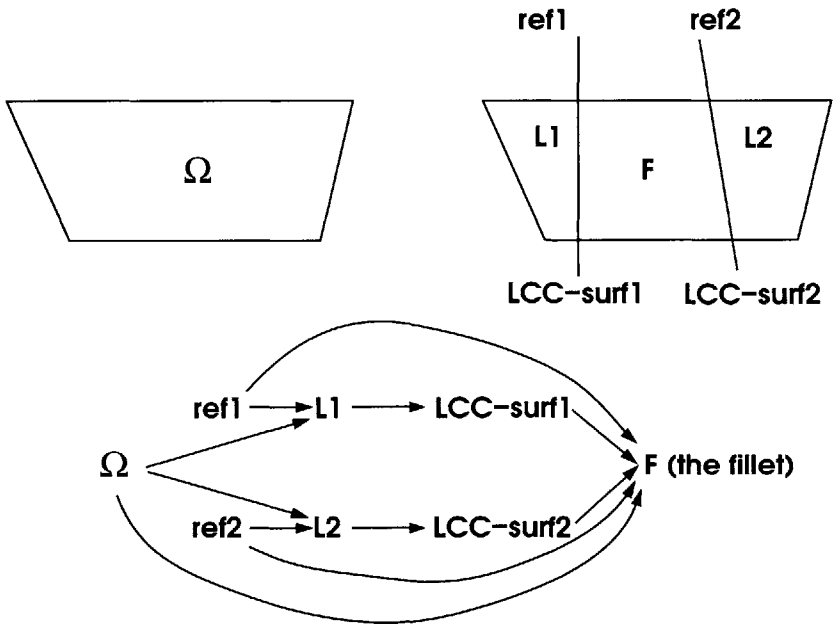


Figure 6-10: The subset of the graph that relates to a LCC fillet feature

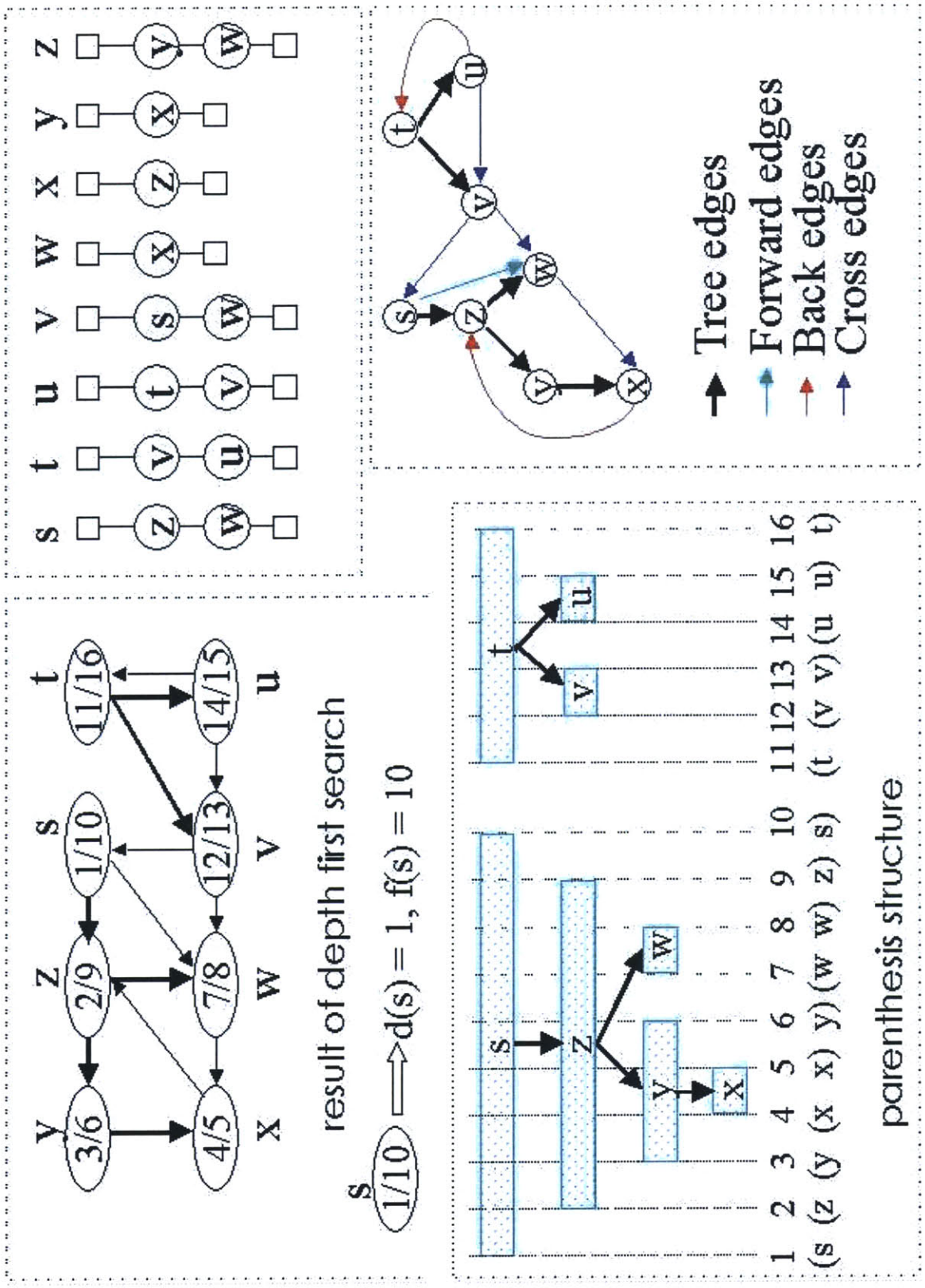


Figure 6-11: The depth-first search of a graph (adapted from [21])

Algorithm 11 bool is_backedge_not_allowable(GraphEdge edgeIn)

identify edgeIn is pure composition type, pure geometric type or both composition and geometric type;
($L1C\_L1S$ or $L1S\_L1C$ or $L1S\_L3$ or $L6\_L1$ are pure composition type; $L1C\_L2$ or $L2\_L1$ are both composition and geometric type; others are pure geometric type)
set all nodes as white;
return SpecDFS_visit(edgeIn.end, target_type, edgeIn.start) to see if there is a path between the two nodes of the edge such that the edges on this path are all in the same category. ▷see Algorithm 12

Algorithm 12 bool SpecDFS_visit(int k, int target_type, int x)

```
set k node as gray;
set returnvalue as false;
for every node t of the graph do
  if k node is adjacent then
    if theEdgeType of edge (k,t) is the same as target_value && node t is node x then
      return true;
    else
      if node t is white then
        returnvalue = SpecDFS_visit(t,target_type,x);
set node k as black;
return returnvalue;
```

Chapter 7

Evaluation of Composition of LCC Object

7.1 Point classification with respect to LCC features

For the evaluation of an LCC object represented with LCC features, it is necessary to classify a query point with respect to the LCC features. The sets of LCC assembly features that are sorted topologically are to be checked. After editing, the LCC assembly features that are edited are the target features.

Point classification is a classic geometric problem. It is often solved by checking the parity of the intersections between a ray shot from the point and the object [54]. In this thesis, an algorithm that utilizes the SolidWorks ray intersection function is developed. The SolidWorks intersection function takes as input a ray and a series of bodies and gives as an output all the intersections points between the ray and the bodies and the types of intersection for each point. The following is the enumerated list of intersection information that the function returns:

swRayPtsResultsFACE;

swRayPtsResultsSILHOUETTE;

swRayPtsResultsEDGE;

swRayPtsResultsVERTEX;

swRayPtsResultsENTER;

swRayPtsResultsEXIT.

The ray takes the form of a base point and direction vector. The basis of the developed algorithm is to find out the closest intersection points with respect to the query point, and using the returned intersection conditions for those intersection points to determine if the query point is interior, coincident or exterior to the geometric boundary of the object. When SolidWorks returns special cases where the ray is coincident with some edge or vertex, the parity checking method is not used. Algorithm 13 is the pseudo-code of the algorithm.

7.2 Composition evaluation at a point, along a ray or on a plane at given resolutions

Composition evaluation at a point is simply calling the evaluation of method of the composition feature of the identified LCC feature that the query point is located in.

7.2.1 Composition evaluation along a given ray at a given resolution

The ray is represented by a starting point X_b and an ending point X_e, with the ending point guiding the direction. $X(t)$ is any point on the ray with parameter t. The resolution is represented by an interval of parameter δt.

$$X(t) = (1 - t) \cdot X_b + t \cdot X_e \tag{7.1}$$

STEP 1 Find the starting point that intersects the bounding box.

STEP 2 Find the end point that intersects the bounding box.

STEP 3 Classify the points at the given resolution with respect to LCC features using the algorithm for classifying all the points with one ray intersection results.

Algorithm 13 Point classification algorithm using SolidWorks ray intersection function

```
1: for each LCC assembly feature in the sorted list from the topological sorter do
2:   if the assembly feature maps to an LCC feature that contains a component or
     face/surface then
3:     if the contained geometry is a face/surface then
4:       check if the query point is incident on the face/surface by calculating the distance;
5:       if Yes then
6:         return the identified LCC assembly feature;
7:     else
8:       get the body of the component;
9:       get the bounding box of the component;
10:      shoot a ray passing the query point from outside of the bounding box from left
         to right (or bottom to top, or back to front).
11:      call SolidWorks intersection function to compute the intersection points;
12:      retrieve the intersection points and intersection condition types;
13:      find the closest intersection points from both sides of the query point;
14:      get the intersection conditions for the closest intersection points;
15:      set left_cnditn01 = (swRayPtsResultsFACE | swRayPtsResultsENTER );
16:      set left_cnditn02 = (swRayPtsResultsSILHOUETTE | swRayPtsResultsENTER
         );
17:      set left_cnditn03 = (swRayPtsResultsEDGE | swRayPtsResultsENTER );
18:      set left_cnditn04 = (swRayPtsResultsVERTEX | swRayPtsResultsENTER );
19:      if cnditn_left meets any of the above left condition criteria then
20:        return the identified LCC assembly feature;
21:      set right_cnditn01 = (swRayPtsResultsFACE | swRayPtsResultsEXIT);
22:      set right_cnditn02 = (swRayPtsResultsSILHOUETTE | swRayPtsResultsEXIT);
23:      set right_cnditn03 = (swRayPtsResultsEDGE | swRayPtsResultsEXIT);
24:      set right_cnditn04 = (swRayPtsResultsVERTEX | swRayPtsResultsEXIT);
25:      if cnditn_right meets any of the above right condition criteria then
26:        return the identified LCC assembly feature;
27:      set special_cnditn01 = (swRayPtsResultsEDGE | swRayPtsResultsENTER |
         swRayPtsResultsEXIT);
28:      set special_cnditn02 = (swRayPtsResultsVERTEX | swRayPtsResultsENTER |
         swRayPtsResultsEXIT);
29:      if cnditn_left== special_cnditn01 or cnditn_right== special_cnditn01 or cn-
         ditn_left== special_cnditn02 or cnditn_right== special_cnditn02 then
30:        if cnditn_right!= (swRayPtsResultsFACE | swRayPtsResultsENTER) then
31:          count the number of intersection to the right of the query point;
32:          if the number is odd then
33:            return the identified LCC assembly feature;
34:      continue;
```

STEP 4 Evaluate the composition for each point: Apply the algorithm for composition evaluation at a point.

Figure 7-1 illustrates the algorithm of evaluation of composition along a ray.

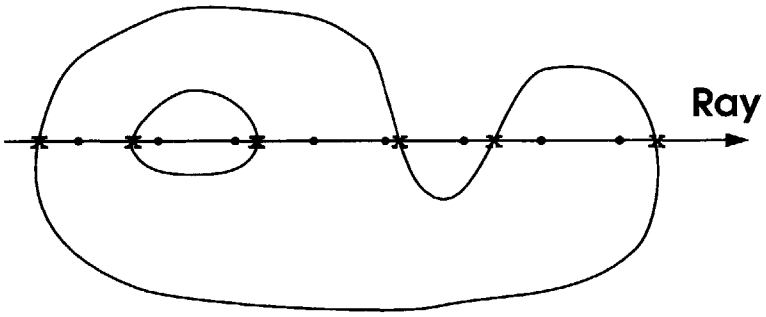


Figure 7-1: Evaluation of composition along a given ray

7.2.2 Composition evaluation on a cutting plane

For the purpose of evaluating compositions of materials, the program will find the composition for the points on a given plane at a given resolution. Here the plane is given in general form

$$A \cdot x + B \cdot y + C \cdot z - D = 0 \tag{7.2}$$

The method is calculating the intersection points of the plane with the bounding box of the model. Using the intersection points coordinates (see Figure 7-2) one can express the plane in parametric form as the following:

$$X(u, v) = X_{b_1} + (X_{b_2} - X_{b_1})u + (X_{b_3} - X_{b_2})v \tag{7.3}$$

After the parametric expression of the plane is obtained, one can repeatedly call the ray casting method to evaluate compositions of the query points on the given plane.

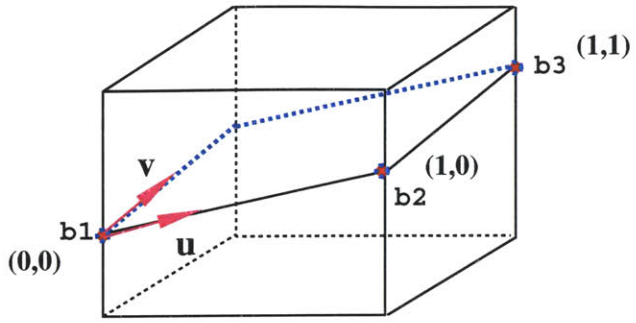


Figure 7-2: Parametric cutting plane

7.3 Intermediate voxel model for visualization or postprocessing for 3DP

As described in Chapter 6, the LCC object data structure includes a 3D grid system as an intermediate voxel model for visualization and postprocessing for 3D Printing. The grid is initialized only when necessary; it's either initialized when Euclidean distance transform is needed or initialized when the output for postprocessing is requested by the user.

The grid object contains a 3D matrix for digital distance transform. It also includes a digital slicer which is an inherited class from LCC_Slicer. It should be noted that each LCC_Slicer's 2D image storage also contains the feature information for each image node, which means image nodes are associated with the LCC features that they belong to. This information is very useful for efficient evaluation of the material compositions.

The digital slicer provides methods to evaluate from the grid system's Euclidean digital distance transform (DT) map the composition of a 2D image parallel to one of the coordinate axes. The grid object also includes a 3D value buffer for outputing the material composition at the grid nodes to data files for postprocessing. The value buffer can be used to store the associate LCC features for each grid node before it is used to store the evaluated composition values.

The grid system also contains parameters such as interval dimensions of the grid, digital distance map origins and grid origins. The grid system provides methods to voxelize the triangulated surfaces of the LCC object as a preprocessing for Euclidean digital distance transform. The methods have been described in the author's master's thesis [43].

7.4 Visualization of outer surfaces of a LCC object

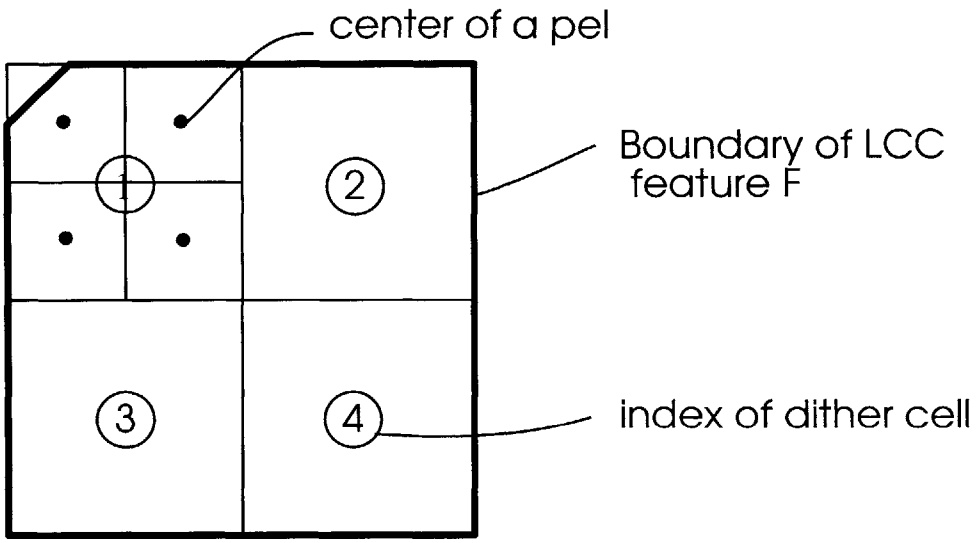
The visualization of outer surfaces of a LCC object is helpful for the user to verify the design or demonstrate the design. Figure 8-3 demonstrates such a visualization. The method is to render the surfaces of each LCC feature. The surfaces are tessellated into triangles and triangles are subdivided to smaller triangles given certain rendering resolution. For each triangle, the three vertices are evaluated for their material composition values using the LCC feature's composition function method. Then, the rendering is done by using the OpenGL routine for smooth blending of triangle vertices. For some of the LCC composition features, special evaluation methods for boundary points are necessary. For example, in the case of a fillet governed by the Laplace equation and solved with the BEM method, the boundary values and derivatives are intermediate results of the solution and then the evaluation of boundary points can be done by retrieving those data from data files.

7.5 Composition evaluation and data output for postprocessing

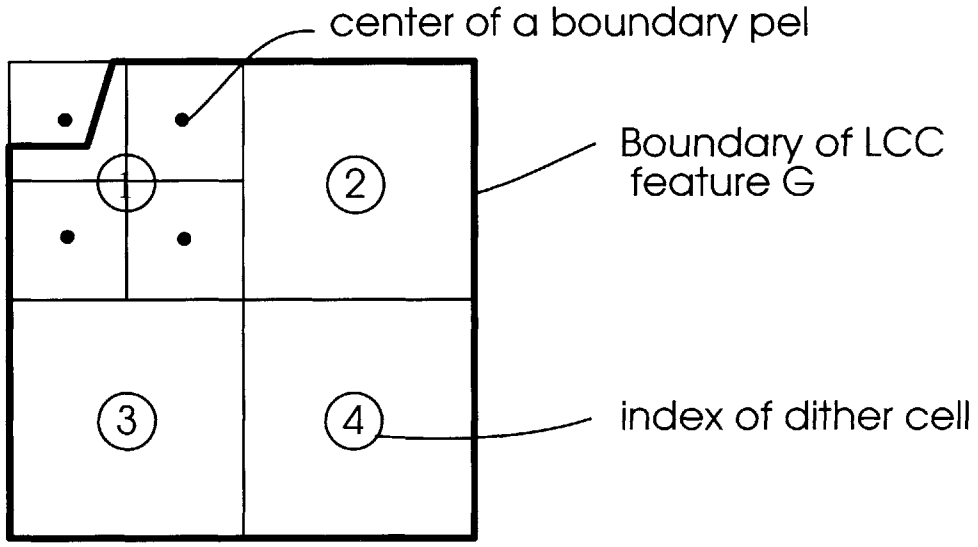
For postprocessing through halftoning, the LCC outputs are the dither cell compositions and the boundary PEL data. Details on postprocessing and halftoning can be found in [19] and [46]. A dither cell consists of $ndmx \times ndmy \times ndmz$ number of PELs and the composition of the cell is evaluated at the center. PELs are the divisions between the adjacent 3D grid at the spacing of px, py, pz along the three Cartesian coordinate axes. The origin of the grid is at the lower left corner of the bounding box of the LCC object. The definition of a boundary cell is: A cell such that some of the center points of its PELs are not interior to any LCC feature. And the definition of a boundary PEL is: A PEL that is part of a boundary cell and its center is interior to some LCC feature. Figure 7-3 demonstrates the concept with examples.

Two data files are used in the postprocessing through halftoning. Their applications can be found in [49].

- Format of the dither data file "DitherData.lcc":



(a) All four dither cells are interior cells and all pels are interior pels



(b) Cell No. 1 is a boundary cell and 3 pels in it are interior pels

Figure 7-3: Dither cells and boundary PELs

PEL size: $px\ py\ pz$ (integers with unit $10^{-6}m$);

Number of dither cells in x, y, z directions respectively: ndx, ndy, ndz;

size of dither matrix: $ndmx, ndmy, ndmz$;

number of materials;

(for each dither cell) the i, j, k indices; material composition value for each material (floating point in range of [0,1]).

- Format of the PEL data file "PELData.lcc":

  PEL size: $px\ py\ pz$ (integers with unit $10^{-6}m$);

  Number of dither cells in x, y, z directions respectively: ndx, ndy, ndz;

  size of dither matrix: $ndmx, ndmy, ndmz$;

  number of boundary dither cells;

  (for each boundary dither cells) the i, j, k indices of the dither cell, number of PELs in this boundary cell, index of each PEL within this cell.

Method of outputing the boundary PELs and dither cells

Algorithm 14 Outputing the boundary PELs and dither cells data

```
1: set the grid resolution at PEL sizes;
2: classify the grid nodes with respect to LCC features and store the information in the
   3D value buffer;
3: if a PEL center is not associated with any LCC feature then
4:    set the PEL as a boundary PEL;
5: set the grid resolution at dither cell sizes;
6: classify the grid nodes with respect to LCC features and store the info in the 3D value
   buffer;
7: evaluate the material compositions for the center of each dither cell.
```

7.6 Time complexity analysis

Time complexity of the point classification algorithm If we denote F_i as the ith LCC feature in the sorted list for point classification, T_i the time spent on classifying the query point with respect to F_i, the total computation time T is $\sum_i T_i$. The time spent on

classifying the point with respect to a feature is the time cost of the ray-body intersection function. Here this item is denoted by $T_{intrs}(i)$. Therefore, $T_i = T_{intrs}(i)$. One can see that $T_{intrs}(i)$ is a function of the geometry of the LCC feature. For many simple shapes, $T_{intrs}(i)$ may be a constant.

Time complexity of classifying a ray of points at a resolution In this case, the algorithm needs to compute the ray-body intersection only once, and then for each point the time for determining the locality is constant. If we denote N_p as the number of the points on the ray, the total time cost is $\sum_i T_{intrs}(i) + c \cdot N_p$, where c is a constant.

Time complexity of classifying rays of points on a plane at set resolutions Similar to the analysis above, if we denote N_r as the number of rays and N_p as the number of points on each ray, the total time cost should be $N_r \cdot \sum_i T_{intrs}(i) + c \cdot N_p N_r$, where c is a constant.

Time complexity of classifying points of a 3D grid at set resolutions This can be easily concluded from the above analysis by considering the grid and layers of 2D rays of points. If we denote N_l as the number of layers, the total time cost is $N_r N_l \cdot \sum_i T_{intrs}(i) + c \cdot N_p N_r N_l$, where c is a constant.

Time complexity for the evaluation of composition The time cost of evaluation of composition on a set of points is the sum of the computation time of the functions that are associated with the LCC features the points belong to. The computation time for Euclidean distance transform is constant once the digital distance map is precomputed through the preprocessor. As for the computation time for the Laplace's equation based blending, each interior point's value is a linear interpolation of the boundary elements values. Therefore, the computation time is of $O(N_b)$, where N_b is the number of the boundary elements of the filleted component.

Chapter 8

Implementation and Numerical Results

8.1 Implementation

A prototype system that includes all the design methods described in this thesis is implemented on an Intel Pentium III PC rated at 1GHz. The system is written in C++ and integrated with SolidWorks [74] system via its Application Programming Interface (API) modules, forming a unified solid modeler for heterogeneous objects. Figure 8-1 shows the user interface of this prototype system. In terms of modules, there exists a material system module, a design module and a visualization and processing module. The material module allows the user to set up an array of materials and assign to each material some property, i.e. the color code. The Design module includes setting default material to all components in the assembly, designing an LCC composition feature for a component or surface in the feature tree of the SolidWorks system, designing an LCC pattern to a component pattern in the feature tree and designing an LCC fillet for a component in the assembly. The composition feature can be designed with any of the methods described previously. The visualization module includes visualization of the cross-section of the LCC object on any user defined plane, visualization of the outer boundary of the object with the material composition color-coded, and processing of the LCC object at user input grid into a data file for lower level postprocessing. The user interface for editing of LCC features is located in the

right mouse button pop-up menu for each component or surface feature in the assembly. In

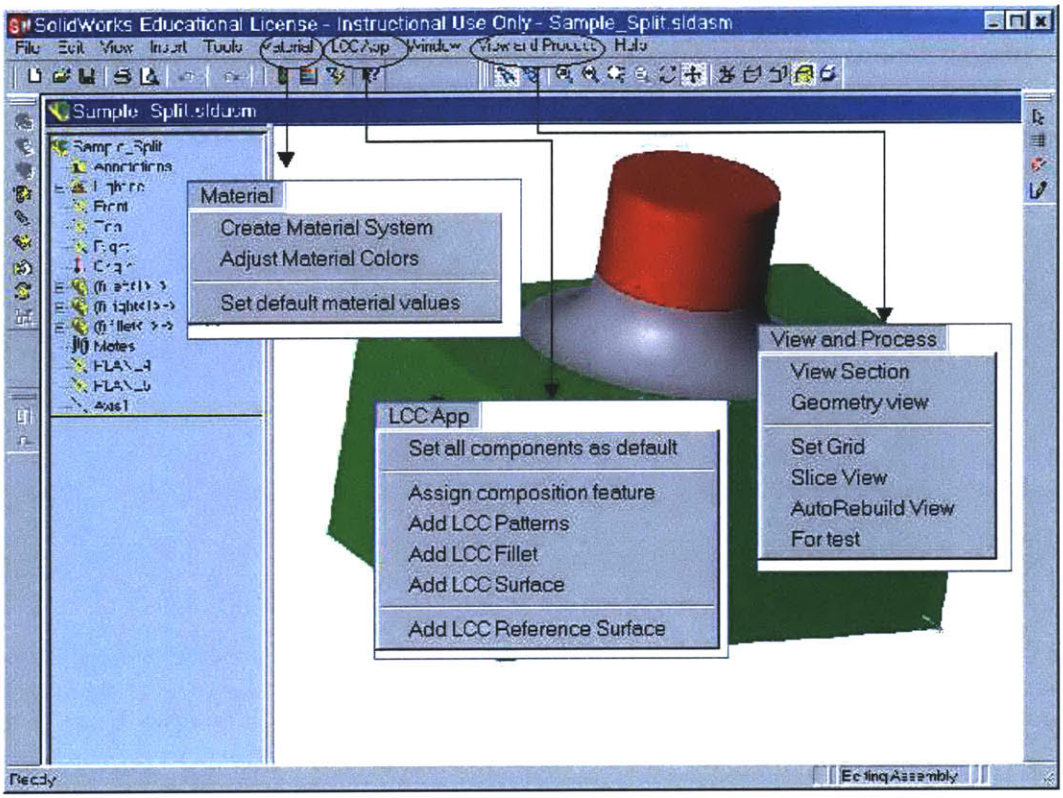


Figure 8-1: Graphical user interface

the following, several examples are presented.

GRIN lens design with distance based composition function: Figure 8-2 shows the design of a GRIN lens. Its geometry is a cylinder with diameter $6.2mm$ and height $0.5mm$. This LCC object is to have one LCC volume feature. The material composition feature has the form $(A + Bsech(Ex))/(C + Dsech(Ex))$, where x is the distance from a point P to the axis divided by the radius of the cylinder, and A = -1184.25, B = 1188.145, C = 92.98925, D = -87.532, and E = 0.080879.

In order to scale the $Max(|Comp|)$ to 1, the data are adjusted to: A = -1184.25, B = 1188.145, C = 66.43268, D = -62.5339, and E = 0.080879.

Tissue scaffold with distance based composition function: Figure 8-3 shows a tissue scaffold that is assigned a composition profile which is a function of the distance to side planar faces which in turn bound the component (the base feature in the part document).

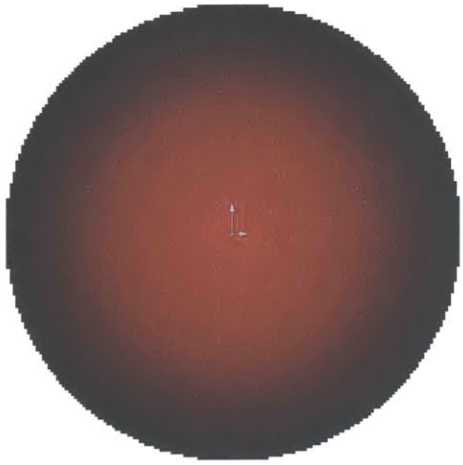


Figure 8-2: GRIN lens

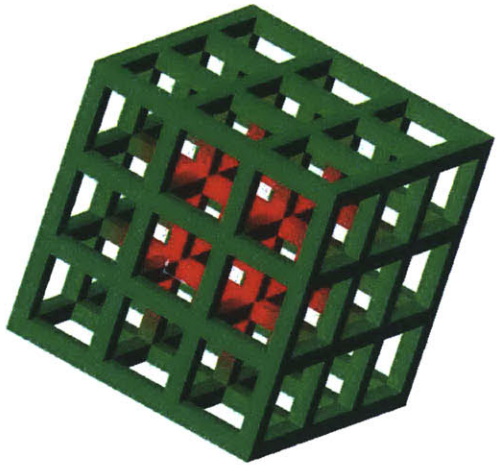


Figure 8-3: Scaffold for tissue engineering

Such a design can promote the growth of tissue into the scaffold.

Tooling part design: using distance function from surface features: The example in Figure 8-4 demonstrates how the composition as a function of distance to different surface features is applied to the tooling part. In Figure 8-4-(b), a material composition profile is assigned as a function of distance to the cooling channel (a sweep feature geometrically) in the tool. Such a design may achieve the local control of porosity in the part and potentially improve the efficiency in tool cooling. In Figure 8-4-(c), the part is assigned a composition profile that is a function of distance to the outer boundary of the part (excluding the cooling channel). The boundary is composed of a series of features that the user built when designing the geometry. This design can be used to control the wear resistance of the tool where near the surface hard phases such as TiC can be printed. In Figure 8-4-(d), the tool is assigned two distance based composition profiles, one to the well, dome and the fillets between them, and the other to the cooling channel. Such a design can facilitate multiple design purposes, i.e. wear resistance control with the first profile and local porosity control with the second profile.

Design of multiple overlapping distance function based composition profiles: The example in Figure 8-5 demonstrates the design result of overlapping composition profiles. Here, there are two geometric surface features in the part, the rectangular base and the cylindrical extrusion. Suppose the user wants to modify the compositions near the two surface features with two different distance based profiles. Then, there is an overlap similar to the example in Section 5.2.3. Here the default composition of the part is 100% yellow, the profile applied to the base grades from 100% red to 100% yellow into the part in the normal direction from the base feature, and the profile applied to the cylindrical extrusion grades from 100% green to 100% yellow into the part in the normal direction from the cylindrical extrusion feature. A design like this allows smooth change of material volume ratio between different profiles.

Pill matrix with LCC pattern: The example in Figure 8-6 shows the design of LCC pattern for structured repetition of a particular LCC feature. Here, a LCC component is inserted into the original pill. Such component is assigned a distance-based composition profile with respect to the boundary of that component. The boundary again is the series

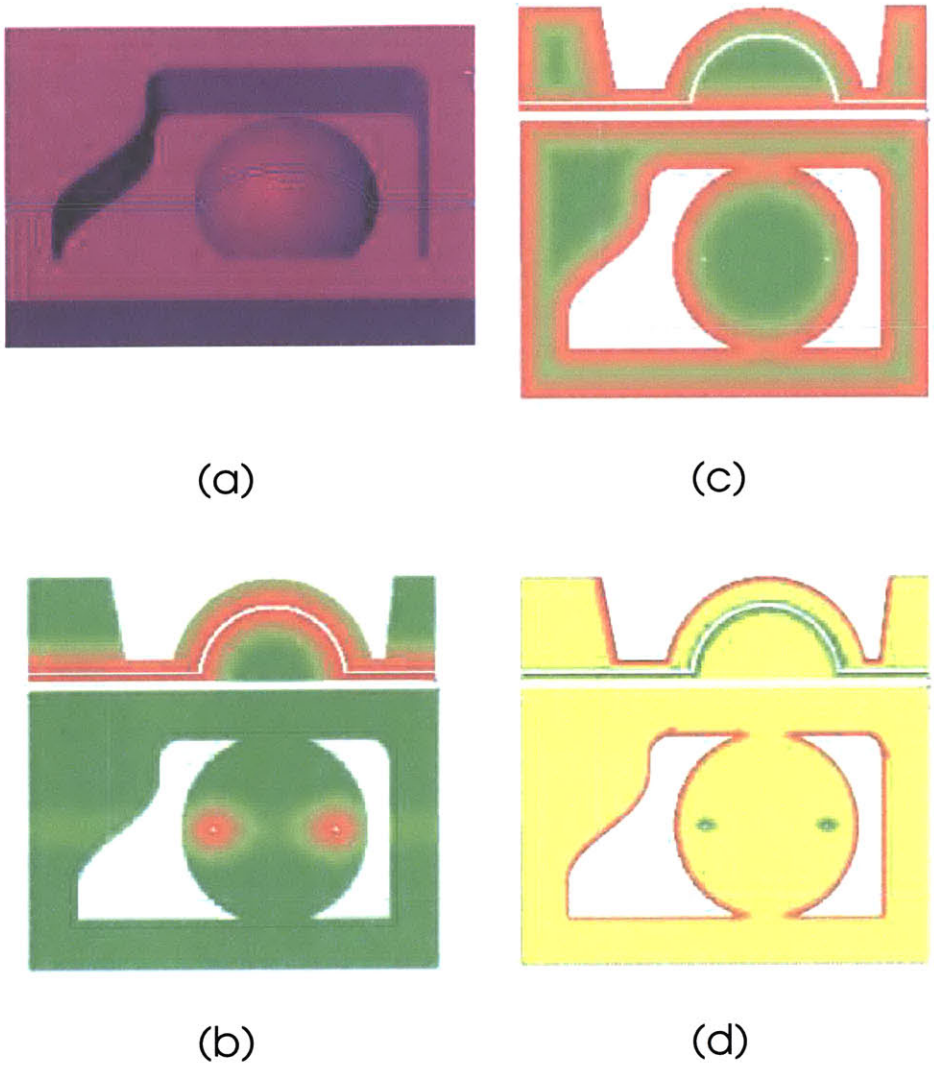


Figure 8-4: Example of tool part design using distance to features method

of features the designer used in designing of the geometry of the component. Then, the component is circularly patterned based on user input parameters. Lastly, the original pill was cut with these inserted LCC components. This design may be used to achieve certain desired functional delivery of drugs.

Design of material fillet using Laplace's equation based blending: The example in Figure 8-7 shows the design of material fillet for adjacent components. The smooth blending is based on solving Laplace's equation with boundary conditions. Here, the example is an assembly of three components A, B and C, where the components are derived

by cutting a single part with parametrized surfaces (two spherical surfaces). The user can assign components A and C as separate LCC features and then assign material fillet on component B. Then, the automatic blending method described in Section 5.3 is executed by our system. Using such a design method, smooth transition between LCC volumes is achieved which may provide better material property than is otherwise achievable.

Editing of geometric feature and composition feature simultaneously: Figure 8-8 shows the simultaneous editing of geometric feature and composition feature with this system. The example in Figure 8-8-(a) shows editing on an LCC feature that is assigned two different distance based composition profiles. In the lower left two sub-figures, the design was edited by changing the length of the base extrude feature, and simultaneously the geometry and the composition are changed. The lower right sub-figure shows the boss feature was removed and the design was updated automatically both for geometry and composition. The example in Figure 8-8-(b) shows editing of the volume fillet of adjacent LCC components. In this example, the radius of the surface fillet was enlarged from $5mm$ to $8mm$. The volume fillet was parametrically designed using the cutting surfaces which are constrained by an equation which is a function of the radius of the surface fillet; therefore the volume fillet was constained by the radius as well. Then, the changes in geometry and composition of the volume fillet were automatically updated after the editing.

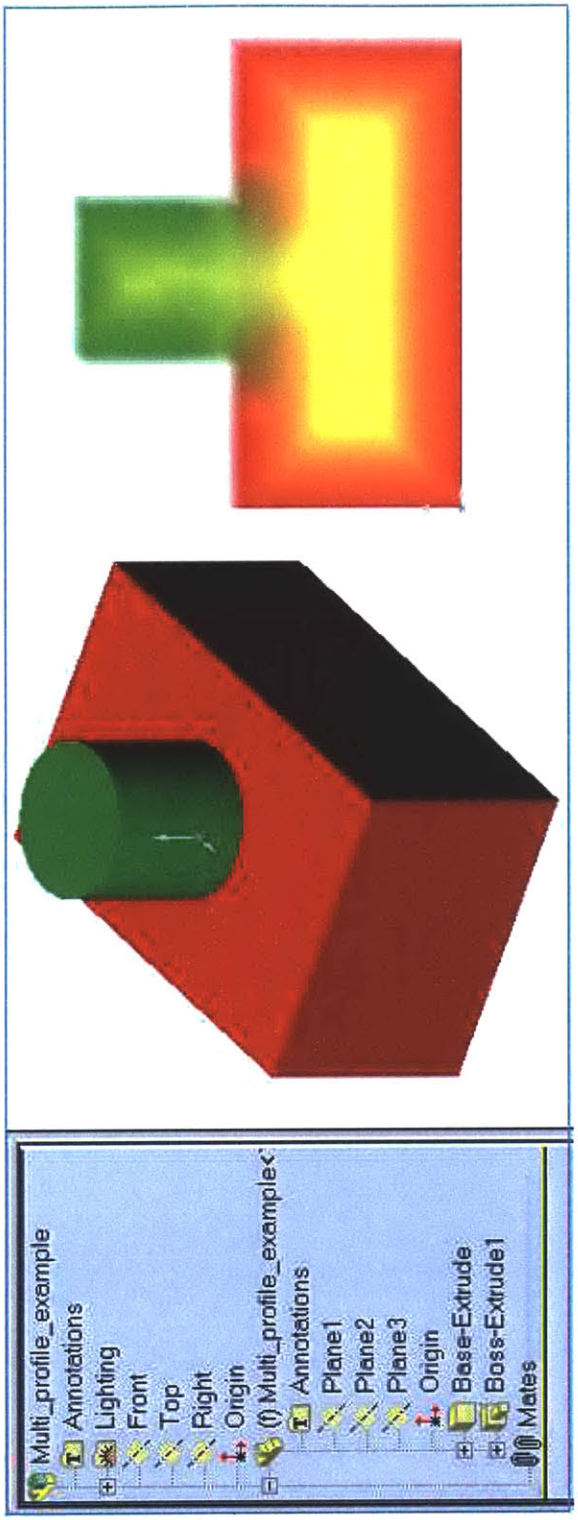


Figure 8-5: Example of multiple overlapping distance function based composition profiles

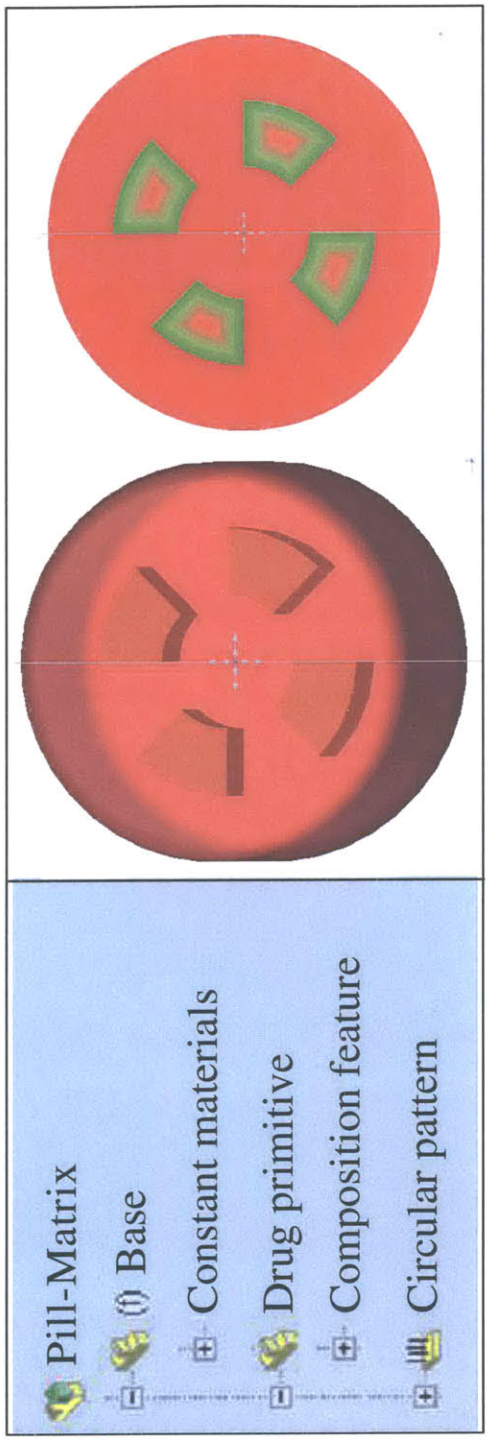


Figure 8-6: Example of pill matrix with LCC pattern

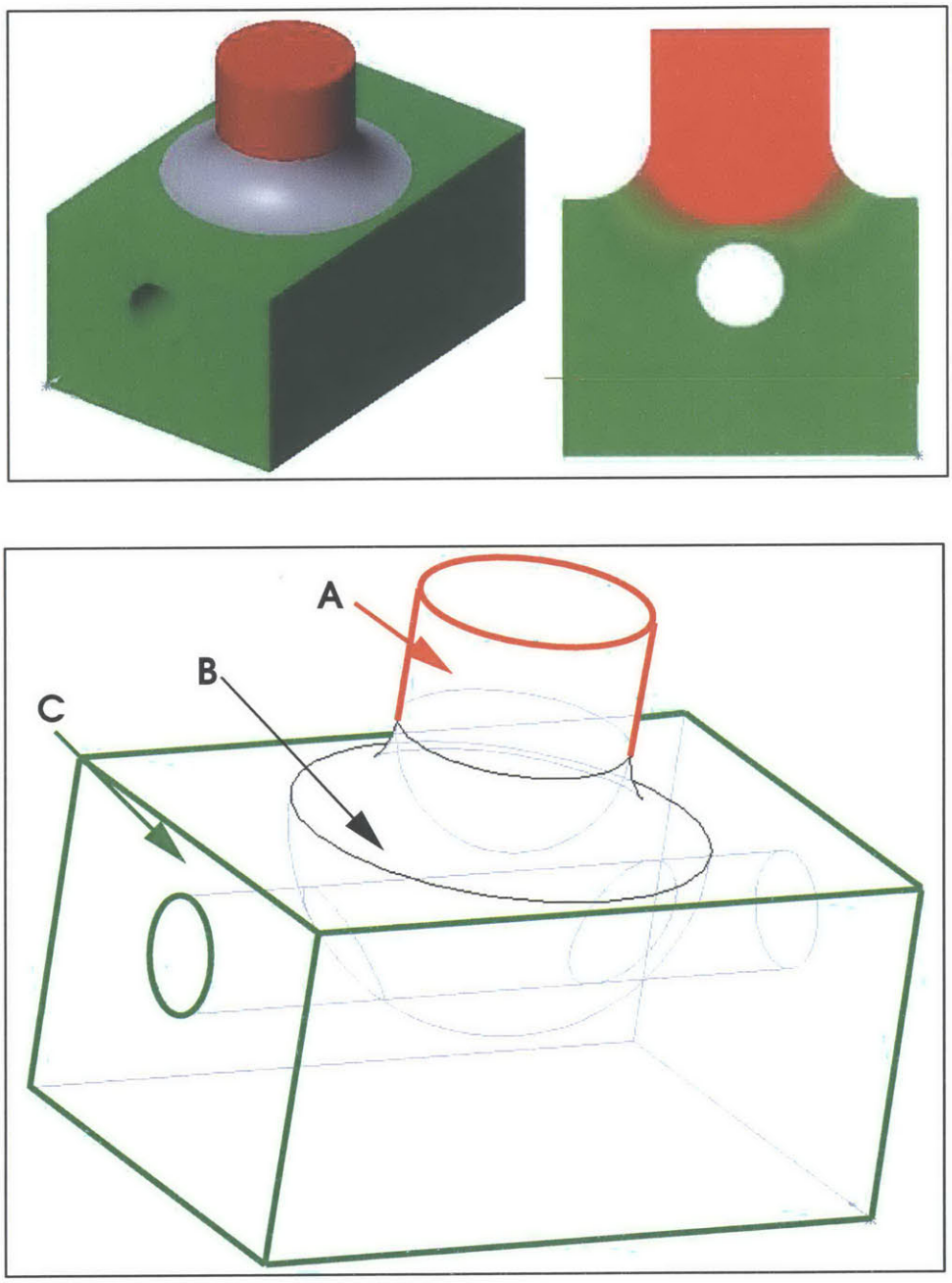
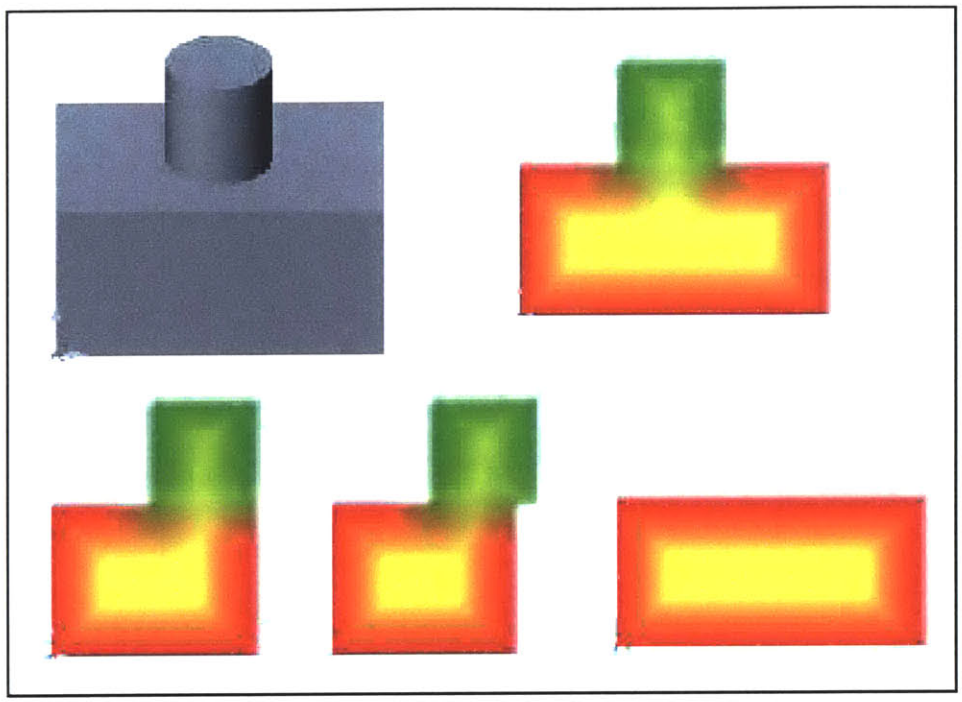
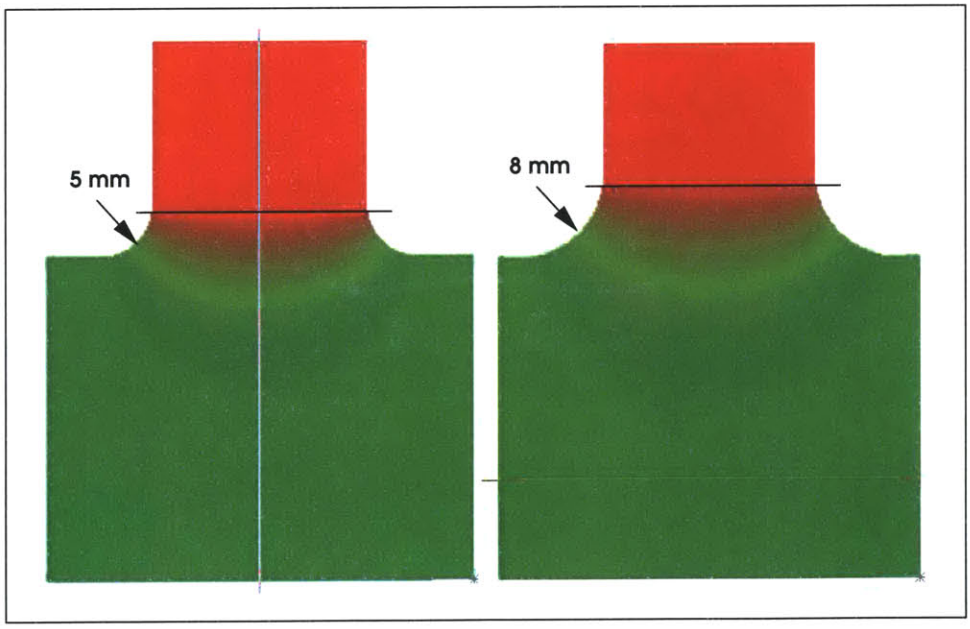


Figure 8-7: Example of material fillet using Laplace's equation based blending



(a)



(b)

Figure 8-8: (a) Edit a multi-profile design; (b) Edit a volume fillet design

8.2 Numerical results

Time performance of Euclidean digital distance transform and evaluation on different models

The following three tables, Table 8.1 to Table 8.3, give experimental running time results of the Euclidean Digital Distance transform and evaluation of the composition on three different models described in Section 5.2.4. Here EDT is the time spent on Euclidean digital distance transform, PT is the time spent on point classification algorithm and RT is the rendering time for two materials.

Number of voxels	*EDT (sec)*	*PT (sec)*	*RT (sec)*
125000	*0.06*	*6.229*	*3.465*
1.00E+06	*0.781*	*27.57*	*28.349*
8.00E+06	*6.419*	*138.249*	*243.661*

Table 8.1: Computation times on example "Cube"

Number of voxels	*EDT (sec)*	*PT (sec)*	*RT (sec)*
17000	*0.01*	*2.533*	*0.662*
134000	*0.13*	*11.126*	*4.176*
1072000	*1.462*	*51.214*	*35.482*

Table 8.2: Computation times on example "Mold tool" in Figure 8-4

Number of voxels	*EDT (sec)*	*PT (sec)*	*RT (sec)*
15625	*0.01*	*0.982*	*2.213*
125000	*0.11*	*4.647*	*9.774*
1000000	*1.532*	*18.667*	*45.335*
8000000	*16.153*	*93.865*	*259.723*

Table 8.3: Computation times on example "Sphere"

Computation time of the blending function using GMRES and LU on example "Sample_split" The Table 8.4 gives the running results of the computation time of the blending function using GMRES and LU method respectively.

Convergence test data on example "Sample_split" The Table 8.5 gives the experimental results on the convergence on the example "Sample_split".

Number of panels	Processing time *with LU solver (sec)*	Processing time *with GMRES solver (sec)*
1074	*77.001*	*34.149*
1245	*122.116*	*45.585*
1479	*215.851*	*63.181*
2271	*684.745*	*146.521*
3396	*2192.713*	*316.765*
4260	*4682.913*	*494.2*

Table 8.4: Computation time of the blending function using GMRES and LU on example "Sample_split" in Figure 8-7

Numerical results on comparison between GMRES methods with or without preconditioning

Table 8.6 gives the numerical results on comparison between GMRES methods with or without preconditioning. The test was conducted on Model "My_Coffeecup" in Figure 5-12 which was tessellated into 2778 elements for the boundary element method.

Numerical results on evaluation time on a 'fillet' example

Table 8.7 gives the running time for evaluation of the "fillet" example. The example is the solid illustrated in Figure 8-1. Here, T_1 is the time cost of the evaluation when the number of query points is 2050, T_2 is the time cost of the evaluation when the number of query points is 8100, and T_3 is the time cost of the evaluation when the number of query points is 50250.

Number of panels	*1074*	*1245*	*1479*	*2271*	*3396*	*4260*
Relative error	*0.012*	*0.004*	*0.003*	*0.0001*	*3.176E-05*	*0*

Table 8.5: Convergence on example "Sample_split" in Figure 8-7

GMRES without preconditioning			*GMRES with preconditioning*		
TOL	*Time on linear eqn. (sec)*	K_{it}	*TOL*	*Time on linear eqn. (sec)*	K_{it}
1.00E-02	*19.99*	*36*	*1.00E-02*	*29.88*	*16*
1.80E-03	*58.88*	*313*	*1.00E-04*	*31.6*	*26*
1.60E-03	*118.63*	*724*	*1.00E-06*	*35.84*	*43*
1.50E-03	*131.68*	*795*	*1.00E-08*	*37.74*	*51*
1.20E-03	*does not converge*		*1.00E-10*	*39.9*	*59*
1.00E-03			*1.00E-12*	*42.08*	*69*

Table 8.6: Numerical comparison between GMRES methods with or without preconditioning

Number of panels	T_1 *(msec)*	T_2 *(msec)*	T_3 *(msec)*
1074	*12157*	*47859*	*301984*
1245	*13991*	*55730*	*352657*
1479	*16724*	*66355*	*415428*
2271	*25297*	*101095*	*646270*
3396	*37814*	*151388*	*955254*

Table 8.7: Time for evaluation on example "test_split02"

Chapter 9

Conclusions and Recommendations

9.1 Conclusions

The major barrier to the wide-spread exploration of the potential of Local Composition Control (LCC) in Solid Freeform Fabrication (SFF) is due to the lack of electronic representations and design tools for objects with LCC. Most CAD research has focused on the representation of 3D geometry of homogeneous objects, on methods and tools for designers to interact with these representations at a high level, and on derivation of machine specific instructions for machining. Current approaches proposed for modeling LCC objects are awkward in editing geometric and material composition information simultaneously. In effect, they permit sequential editing (i.e., first of geometry and then composition), which is not flexible and limits the designer's options. Current LCC models are also limited to low level data and operators, and do not allow for the symbolic representation of the designer's intent with respect to composition. In addition, design changes cannot be efficiently propagated. In order to address these limitations, this thesis builds on the concept of feature-based design (FBD) and extends it from a geometric domain to simultaneous editing of material and geometric features.

This thesis has identified and formalized the concept of LCC features. The classes of LCC features include those based on volume, transition, pattern, and (user-defined) surface features. Methods for LCC feature creation and editing were developed. Specifically, material composition functions such as functions parameterized with respect to distance or

distances to user-defined geometric features, and functions that use Laplace's equation to blend smoothly various boundary conditions including values and gradients of the material composition on the boundaries were developed. The Euclidean digital distance transform and the Boundary Element Method were employed for the efficient computation of composition functions. In addition, the General Minimization of Residual Method was employed as an appropriate iterative method for solving the resulting linear equation system. Theoretical and experimental complexity, accuracy and convergence analyses were presented as well.

With such a feature-based scheme the efficient and robust evaluation of a LCC object was done at different levels of resolution for both visualization and fabrication purposes. An unevaluated exact representation for the geometry and composition was maintained for as long as possible along the information pathway. Therefore, a high level codification of the design useful for data exchange in a general setting not associated with a specific SFF process was provided. The system's multi-level architecture and model manager provide more controls on the validity of a solid with LCC, and efficiency in updating the design and evaluation.

In this thesis, examples were also presented in order to demonstrate the usefulness of such a system in exploring the potential applications in SFF with LCC. The examples included tissue scaffold, tool part with local control, drug delivery device, GRIN lens, material fillet, etc.

9.2 Recommendations

Within the scope of the feature-based design, greater variety of composition design features is needed for the envisioned potential applications. For example, different types of blending of material composition are needed. B-spline basis functions are an attractive set of mathematical functions that may be used for blending. It can be formed as a dependent feature of LCC surface/faces features. In terms of mathematical representation of the composition profiles, a more general functional form, such as the spread sheet type of expression is desirable.

Another possible direction is bringing the design tool to a higher level to enable more design methods to capture the users' real world design intent that are often physical, descriptive or even aesthetic. In order to produce Functionally Gradient Material (FGM) solids successfully through SFF processes, it is also necessary for the design system to be adjusted according to design rules derived from the process limits. In order to evaluate the functionality of a FGM part, it is necessary to analyze the physical properties as functions of material composition. The design system may provide the basic information, such as minimum or maximum material gradient, iso-surfaces etc. or more advanced algorithms for analyzing physical properties directly. Finite-Element Analysis can be integrated in the system for design and redesign of heterogeneous objects. The meshing algorithm can be improved with the rich feature information in the system and an adjacency graph of the LCC features.

For the sake of effective design and redesign, good visualization is also necessary. Iso-surface extraction can be useful for verifying the design.

In addition, general adaptive subdivision of a solid model may be needed to reduce the model size while keeping the accuracy of the intermediate model, where the subdivision of the models can be either structural (for example a grid) or non-structural and subjected to different applications.

Postprocessing algorithms play an important role as an interface between the ideal CAD model with FGM and its machine instructions for LCC with SFF. Improvement on the accuracy in approximation of both the geometry and composition can be pursued further.

Future research on design of FGM may also be oriented to heterogeneous model exchange standards, distribution and fabrication through data exchange via the Internet.

Appendix A

Algorithms

A.1 Algorithm: Euclidean Digital Distance Transform

This algorithm is adapted from [68].

Input picture: $F = f_{ijk}, f_{ijk} = 0$ or 1.

Output picture (distance transformation): $F = f_{ijk}$.

L, M, N sizes of pictures (numbers of rows, columns and planes).

α = grid length along j/ grid length along i.

β = grid length along k/ grid length along i.

$buff(n)$: one-dimension work array with the size n.

$int(x)$: function to convert the data type from the real type to the integer type.

$min(x, y)$:function to select smaller of x and y.

$sqrt(x)$: function to calculate the square root of x.

(Step 3) Same as Step 2 except that the *FOR* loops are respectively over j, i and k, and the α should be β.

Algorithm 15 Step 1

```
    forward scan
    for k = 1 → N do
      for j = 1 → M do
        df = L;
        for i = 1 → L do
6:        if f_ijk ≠ 0 then
            df = df + 1;
          else
            df = 0;
            f_ijk = df^2;
    backward scan
12: for k = 1 → N do
      for j = 1 → M do
        db = L;
        for i = L → 1 do
          if f_ijk ≠ 0 then
            db = db + 1;
18:       else
            db = 0;
            f_ijk = min(f_ijk, db^2);
```

Algorithm 16 Step 2

```
    for k = 1 → N do
      for i = 1 → L do
        for j = 1 → M do
          buff(k) = f_ijk;
        Forward scan
        a = 0;
 7:     for j = 2 → M do
          if a > 0 then
            a = a − 1;
          if buff(j) > buff(j − 1) + α² then
            b = (buff(j) − buff(j − 1) − α²)/(2α²);
            if (j + b) > M then
              b = M − j;
14:         for n = a → b do
              m = buff(j − 1) + (n + 1)² × α²;
              if buff(j + n) ≤ m then
                goto 38 ;
              if m < f_i(j+n)k then
                f_i(j+n)k = m;
            a = b;
21:       else
            a = 0;
        Backward scan
        a = 0;
        for j = M − 1 → 1 do
          if a > 0 then
            a = a − 1;
28:       if buff(j) > (buff(j + 1) + α²) then
            b = (buff(j) − buff(j + 1) − α²)/2(α²);
            if (j − b) < 1 then
              b = j − 1;
            for n = a → b do
              m = buff(j + 1) + (n + 1)²α²;
              if buff(j − n) ≤ m then
35:             goto ;
              if m < f_i(j−n)k then
                f_i(j−n)k = m;
            a = b;
          else
            a = 0;
```

A.2 Algorithm: Left-Preconditioned GMRES Method with x_o an Initial Guess

This algorithm is adapted from [63].

Algorithm 17 Solving linear system $Ax = b$ with left-preconditioned GMRES method

1: Solve r_o from $Mr_o = b - Ax_o$;
2: $v_1 = r_o/\beta$ where $\beta = ||r_o||_2$;
3: **for** $j = 1, \cdots, m$ **do**
4: Solve w from $Mw = Av_j$;
5: **for** $i = 1, \cdots, j$ **do**
6: $h_{i,j} = (w, v_i)$;
7: $w = w - h_{i,j}v_i$;
8: $h_{j+1,j} = ||w||_2$;
9: **if** $h_{j+1,j} = 0$ **then**
10: set $m = j$ and go to 12;
11: $v_{j+1} = w/h_{j+1,j}$;
12: Define $V_m = [v_1, v_2, \cdots, v_m]$, $\bar{H}_m = [h_{i,j}]_{1 \le i \le j+1; 1 \le j \le m}$;
13: Compute y_m which minimizes $||\beta e_1 - \bar{H}_m y||_2$;
14: $x_m = x_o + V_m y_m$

In other words the solution in the *i-th* iterate of GMRES is constructed as

$$x^{(i)} = x_o + y_1 v^{(1)} + \cdots + y_i v^{(i)} \tag{A.1}$$

where y_i is determined to minimize the residual norm $||b - Ax^{(i)}||$.

A.3 Algorithm: Depth First Search for Topological Sort

This algorithm is adapted from [21].

Algorithm 18 Depth First Search: DFS(G)

```
1: for each vertex u in V[G] do
2:    color[u] = white;
3: time = 0;
4: for each vertex u in V[G] do
5:    if color[u] == white then
6:       DFS-Visit(u);
```

Algorithm 19 Depth First Search Visit: DFS-Visit(u)

```
1: color[u] = gray;
2: time = time+1;
3: d[u] = time;
4: for each v in Adj[u] do
5:    if color[v] == white then
6:       DFS-Visit(v);
7:    if color[v] == gray then
8:       Set edge (u,v) as back;
9: color[u] = black;
10: time = time+1;
11: f[u] = time;
12: insert u to result list
```

where V[G] is the vertex set of graph G.

Bibliography

[1] 3D Systems, Inc. *Stereolithography Interface Specification*, July 1988.

[2] S. Ashley. Rapid Prototyping System, *Mechnical Engineering. NY: American Society of Mechanical Engineers*, 113(4):34-43, April 1991.

[3] American National Standards Institute. *Product Data Exchange Using STEP (PDES) Part 42, Integrated generic resources: geometric and topological representation.* Fairfax, VA, February 1995.

[4] R. Bidarra, K. J. Kraker, and W. F. Bronsvoort. Representation and management of feature information in a cellular model, *Computer-Aided Design*, 30(4):301–313, 1998.

[5] R. Bidarra, A. Idri, A. Noort and W. F. Bronsvoort. Declarative user-defined feature classes, *Proceedings of DETC'98, ASME Design Engineering Technical Conferences,* September, 1998, Atlanta, Georgia.

[6] R. Bidarra and W. F. Bronsvoort. Semantic feature modeling, *Computer-Aided Design*, 32(3):201–225, March 2000.

[7] A. Biswas, V. Shapiro and I. Tsukanov. Heterogeneous material modeling with distance fields, *Computer-Aided Geometric Design*, 21(3):215-242, March 2004.

[8] M. I. G. Bloor and M. J. Wilson. Using partial differential equations to generate free-form surfaces, *Computer-Aided Design*, 22(4):202–212, 1990.

[9] G. Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27: 321-345, 1984.

[10] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. Boundary Element Techniques. *Springer-Verlag*, 1984.

[11] G. Brunetti and A. Stork. Product-centered intuitive 3D interaction for feature-based parametric assembly modelling. *Proceedings of the ProSTEP Science Days'98,* pp. 17-18, 1998.

[12] G. Brunetti and B. Golob. A feature-based approach towards an integrated product model including conceptual design information. *Computer-Aided Design*, 32:877–887, 2000.

[13] V. Capoyleas, X. Chen, and C. M. Hoffmann. Generic naming in generative, constraint-based design, *Computer-Aided Design*, 28(1):17–26, 1996.

[14] H. S. Caslaw and J. C. Jaeger. Conduction of Heat in Solids. *Oxford University Press*, 1959.

[15] V. Chandru, S. Manohar, and C. E. Prakash. Voxel-based modeling for layered manufacturing. *IEEE Computer Graphics and Applications*, 15(6):42–47, November 1995.

[16] X. Chen and C. M. Hoffmann. Toward feature attachment, *Computer-Aided Design*, 27(9):695–702, 1995.

[17] X. Chen and C. M. Hoffmann. On editability of feature-based design, *Computer-Aided Design*, 27(12):905–914, 1995.

[18] W. Cho, E. M. Sachs, N. M. Patrikalakis, H. Liu, H. Wu, T. R. Jackson, C. C. Stratton, J. Serdy, M. J. Cima, and R. Resnick. Methods for distributed design and fabrication of parts with local composition control. *Proceedings of the 2001 NSF Design and Manufacturing Grantees Conference*, Tampa, FL, USA, January 2001.

[19] W. Cho, E. M. Sachs, N. M. Patrikalakis and D. E. Troxel. A dithering algorithm for local composition control with Three-Dimensional Printing. *Computer-Aided Design*, 35(9):851-867, August 2003.

[20] Component Object Model, http://www.microsoft.com

[21] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms,* MIT Press, Cambridge, MA, 1990.

[22] U. Cugini. Feature-based assembly for aeronautics. *CAD Tools and Algorithms for Product Design, P. Brunet, C. Hoffmann and D. Roller (Eds.)*, pp. 31–46, Springer, 2000.

[23] H. Dedhia, V. Pherwani, J. Shah. Dynamic interfacing of applications to geometric modelers via neutral protocol. *Computer-Aided Design*, 29(12): 811-824, 1997.

[24] D. Deneux. Introduction to assembly features: an illustrated synthesis methodology. *Journal of Intelligent Manufacturing*, 10: 29-39, 1999.

[25] V. Frayssé, L. Giraud and S. Gratton. A Set of GMRES Routines for Real and Complex Arithmetics. *Technical report*, CERTACS, TR/PA/97/49, Toulousse, France. http://www.cerfacs.fr/algor/Softs

[26] D. C. Gossard, R. P. Zuffante, and H. Sakurai. Representing dimensions, tolerances, and features in MCAE systems. *IEEE Computer Graphics and Applications*, 8(2):51-59, 1988.

[27] C. M. Hoffmann and R. Joan-Arinyo. Erep-an editable high-level representation for geometric design and analysis, *Geometric Modeling for Product Realization, P. Wilson, M. Wozny and M. Pratt (Eds.)*, North-Holland, Amsterdam, pp. 129-164, 1993.

[28] C. M. Hoffmann and R. Joan-Arinyo. On user-defined features, *Computer-Aided Design*, 30(5):321-332, April 1998.

[29] W. van Holland and W. F. Bronsvoort. Assembly features in modeling and planning. *Robotics and Computer Integrated Manufacturing*, 16:277-294, 2000.

[30] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*, A. K. Peters, Wellesley, MA, 1993.

[31] S. Holzner. Microsoft Visual C++ 5. *SYBEX Inc.*, 1997.

[32] 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.

[33] T. R. Jackson. *Analysis of Functionally Graded Material Object Representation Methods,* PhD thesis, Massachusetts Institute of Technology, January 2000. (http://web.mit.edu/tdp/www/info-flow/publications/)

[34] 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.

[35] 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*, pp. 259–266, Austin, Texas, August 10-12 1998. The University of Texas.

[36] T. R. Jackson, W. Cho, N. M. Patrikalakis and E. M. Sachs. Memory analysis of solid model representations for heterogeneous objects. *ASME Transactions, Journal of Computing and Information Science in Engineering*, 2(1): 1-10, March 2002.

[37] F. John. Partial Differential Equations, 4th edition. *Springer-Verlag*, 1982.

[38] W. E. Katstra, R. D. Palazzolo, C. W. Rowe, B. Giritlioglu, P. Teung, and M. J. Cima. Oral dosage forms fabricated by Three-Dimensional Printing^{TM}. *Journal of Controlled Release*, 66(1):1–9, May 2000.

[39] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *Computer*, 26(7):51–64, July 1998.

[40] C. T. Kelley. Iterative Methods for Linear and Nonlinear Equations. *SIAM*, 1995.

[41] V. Kumar and D. Dutta. An approach to modeling and representation of heterogeneous objects. *ASME Transactions, Journal of Mechanical Design*, 120:659–667, December 1998.

[42] H. Liu, W. Cho, T. R. Jackson, N. M. Patrikalakis, and E. M. Sachs. Algorithms for design and interrogation of functionally gradient material objects. *Proceedings of 2000 ASME DETC/CIE, 26-th ASME Design Automation Conference,* September, 2000, Baltimore, Maryland, USA. p.141 and CDROM, NY:ASME, 2000.

[43] H. Liu. *Algorithms for Design and Interrogation of Functionally Graded Material Solids*, Master's Thesis, Massachusetts Institute of Technology, February 2000. (http://czms.mit.edu/cho/3dp/onr/report/0700/hl-thesis.pdf)

[44] H. Liu, T. Maekawa, N. M. Patrikalakis, E. M. Sachs, and W. Cho. Methods for Feature-Based Design of Heterogeneous Solids *Computer-Aided Design*, 2003. In press.

[45] H. Liu, N. M. Patrikalakis, E. M. Sachs, T. Maekawa, W. Cho and C. C. Stratton. A design and post-processing system for local composition control in solid freeform fabrication. *Proceedings of the 2004 NSF Design, Service and Manufacturing Grantees and Research Conference*, Dallas, Texas, USA. January 2004.

[46] H. Liu, T. Maekawa, C. Stratton, N. M. Patrikalakis, E. M. Sachs and W. Cho. Testing of the Postprocessing Methods for Heterogeneous Solids. MIT Design Laboratory Memorandum 03-1, February 2003.

[47] H. Liu, T. Maekawa, C. C. Stratton, N. M. Patrikalakis, E. M. Sachs, and W. Cho. LCC Objects via Postprocessing through Halftoning. MIT Design Laboratory Memorandum 03-2, February 2003.

[48] H. Liu, T. Maekawa, N. M. Patrikalakis, E. M. Sachs. User's Guide: A Feature-Based Design System for Solids with Local Composition Control. MIT Design Laboratory Memorandum 03-4, February 2003.

[49] T. Maekawa, C. C. Stratton, W. Cho, H. Liu, E. M. Sachs and N. M. Patrikalakis. User's Manual for Post-Processing Software of Local Composition Control in 3D Printing. MIT Design Laboratory Memorandum 03-3, March 2003.

[50] M. Mäntylä. *An Introduction to Solid Modeling,* Computer Science Press, Rockville, Maryland, 1988.

[51] W. Martin and E. Cohen. Representation and extraction of volumetric attributes using trivariate splines: a mathematical framework. *Proceedings of the 6th ACM Symposium on Solid Modeling and Applications*, In D. C. Anderson and K. Lee (Eds.), pp. 234-240, 2001.

[52] R. Merz, F. B. Prinz, K. Ramaswami, M. Terk, and L. Weiss. Shape Deposition Manufacturing. *Proceedings of the Solid Freeform Fabrication Symposium*, pp. 1-8, University of Texas at Austin, August 1994.

[53] E. H. Nielsen, J. R. Nixon, and G. E. Zinsmeister. Capturing and using designer intent in a design with features system. *Proceedings of Design Theory and Methodology*, D-E Vol. 31, ASME, pp. 95–102, 1991.

[54] J. O'Rourke. Computational Geometry in C. *Cambridge University Press*, 1993.

[55] Parasolid. (http://www.parasolid.com/), Cambridge, UK.

[56] R. Paris and J. Canas. Boundary Element Method. *Oxford University Press*, 1997.

[57] S. Park, R. H. Crawford and J. J. Beaman. Volumetric multi-texturing for functionally gradient material representation. *Proceedings of the 6th ACM Symposium on Solid Modeling and Applications*, In D. C. Anderson and K. Lee (Eds.), pp. 216-224, 2001.

[58] J. Pegna and A. Safi. CAD modeling of multi-modal structures for free-form fabrication, 1998. In D. L. Bourell et al, editor, *Solid Freeform Fabrication Symposium*, Austin, Texas, August, 1998. The University of Texas.

[59] M. J. Pratt, A. D. Bhatt, D. Dutta, K. W. Lyons, L. Patil and R. D. Sriram. Progress towards an international standard for data transfer in rapid prototyping and layered manufacturing. *Computer-Aided Design*, 34(14): 1111-1121, December 2002.

[60] M. J. Pratt. Synthesis of an optimal approach to form feature modeling. *Proceedings of the ASME 1988 Computers in Engineering Conference,* vol. 1, pp. 263–274, New York, 1988.

[61] X. Qian and D. Dutta. Design of heterogeneous turbine blade, *Computer-Aided Design*, 35: 319-329, 2003.

[62] J. R. Rossignac. Issues on feature-based editing and interrogation of solid models, *Computers and Graphics*, 14(2):149–172, 1990.

[63] Y. Saad. Iterative methods for sparse linear systems. *Thomson Publishing: PWS/ITP*, 1996.

[64] 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*, 40(5):1232–1247, May 2000.

[65] 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*, pp. 35–36. Arlington, VA: NSF, January 1998.

[66] E. M. Sachs, J. Haggerty, M. J. Cima, and P. Williams. Three-Dimensional Printing Techniques, U.S. Patent No. 5204055, April 20, 1993.

[67] E. M. Sachs, M. J. Cima, P. Williams, D. Brancazio, and J. Cornie. Three dimensional printing: rapid tooling and prototypes directly from a CAD model, *Journal of Engineering for Industry,* 114(4):481-488, November 1992.

[68] T. Saito and J. Toriwaki. New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications, *Pattern Recognition*, 27(11):1551–1565, 1994.

[69] O. W. Salomons, F. J. A. M. van Houten and H. J. J. Kals. Review of research in feature-based design, *Journal of Manufacturing Systems*, 12(2):113–132, 1993.

[70] J. Shah and M. Mäntylä. *Parametric and Feature-Based CAD/CAM*, John Wiley, Inc., 1995.

[71] J. Shah and M. T. Rogers. Assembly modeling as an extension of feature-based design, *Research in Engineering Design*, 5:218-237, December 1993.

[72] K. Shin and D. Dutta. Constructive representations for heterogeneous objects. *ASME Transactions, Journal of Computing and Information Science in Engineering*, 1(3): 205-217, September 2001.

[73] Y. K. Siu and S. T. Tan. Source-based heterogeneous solid modeling. *Computer-Aided Design*, 34:41-55, 2002.

[74] SolidWorks^{TM} (http://www.solidworks.com/), Concord, MA.

[75] D. G. Ullman. The evolution of function and behaviour during mechanical design. *Proceedings Design Theory and Methodology*, D-E Vol. 53, ASME, pp. 91–103, 1993.

[76] 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*, pp. 3-36, Elsevier Science Publishers, Holland, 1986.

[77] 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*, pp. 37-66, Elsevier Science Publishers, Holland, 1986.

[78] H. Wu, E. M. Sachs, N. M. Patrikalakis, D. Brancazio, J. Serdy, T. R. Jackson, W. Cho, H. Liu, M. J. Cima, and R. Resnick. Distributed design and fabrication of parts with local composition control. *Proceedings of the 2000 NSF Design and Manufacturing Grantees Conference*, Vancouver, BC, Canada, January 2000.

[79] H. B. Xu. Numerical study of fully non-linear water waves in three dimensions. *PhD Thesis*, MIT, Cambridge, MA, USA, 1992.

[80] M. Xue, H. Xu, Y. Liu and D. K. P. Yue. Computation of fully-nonlinear three-dimensional wave-wave and wave-body interactions. Part I: Dynamics of steep three-dimensional waves. *Journal of Fluid Mechanics*, 438: 11-39, 2001.

[81] E. Zauderer. Partial Differential Equations of Applied Mathematics. John Wiley & Sons, Inc., NY, 1983.