A Surface Texture Modeling System for Solid Freeform Fabrication

by
John G. Nace

B.S. Mathematical Sciences
Pennsylvania State University
1985

Submitted to the Department of
Mechanical Engineering in partial Fulfillment of
the Requirements for the
Degree of

Master of Science

at the

Massachusetts Institute of Technology
September 1997

Signature of Author_____
Department of Mechanical Engineering
July 30, 1997

Certified by_____
Prof. Emanuel M. Sachs
Professor, Mechanical Engineering
Thesis Supervisor

Certified by_____
Prof. Duane Boning
Associate Professor, Electrical Engineering

Accepted by_____
Ain A. Sonin
Chairman, Graduate Committee

A Surface Texture Modeling System for Solid Freeform Fabrication

by
John G. Nace

Submitted to the Department of Mechanical Engineering
on July 30, 1997
in partial fulfillment of the requirements for the degree of
Master of Science of Mechanical Engineering

Abstract

Solid Freeform Fabrication, SFF, is a set of manufacturing processes that fabricates parts as a bonded stack of individual layers. The Three Dimensional Printing process, 3DP™ process, is an SFF technology developed at MIT. It builds layers by ink jet printing binder onto the surface of a bed of powder. The bed of powder is lowered and fresh powder is spread onto the bed. As subsequent cross sections of the part are printed, the part exists, submerged in the powder bed.

Access to the individual layers as they are fabricated gives access to the interior structure of the part. This approach allows the part to have high geometric complexity.

In this work a designer centric Computer Aided Design system is proposed to allow the interactive creation of functional surface texture on mechanical parts. This system is structured to behave like a VLSI CAD system, which offers substantial process capabilities.

The requirements for a Mechanical CAD, MCAD, system to behave like VLSI CAD are determined to be: 1. That the informational model of the unit cell of texture be separable into distinct logical subsets.2. That manipulations on either subset not violate the logical consistancy of the other subset.

This thesis shows that geometric dimensions and tolerances carry the essential information of the model of a unit cell of functional texture.

A variety of Unit Cell editors are evaluated according to their ability to meet the desired system criteria. A tool, Swiss Solid Geometry, SSG, for the design of unit cells of functional texture is developed, that fullfills requirement #1. SSG is an approach to MCAD modeling that combines geometric primitives in the manner of Constructive Solid Geometry, however the primitives of SSG, are different. They consist of simple objects such as lines, but includes the spatial envelope around them of a fixed offset. Also, they are used to represent both positive and negative regions of space.

The placement of the individual replications is established by a mesh, that covers the intended 3D surface region. A meshing algorithm is developed that regularizes the mesh by directly utilizing the dimensional tolerances specified in the process of Unit Cell design. The geometric dimensions are instantiated as standalone geometric entities that push and pull on the nodes of the mesh in order to bring their length into dimensional tolerance. This method fullfills requirement #2, and it is implemented into a CAM software called Vari_4.

The modularity of the CAM software, Vari_4, is described in detail.

Thesis Supervisor: Professor Emanuel Sachs
Title: Professor of Mechanical Engineering

Acknowledgment

Jeanette Marie Nace, my wife, has supported me throughout this endeavor. She has carried the heavy burdens of raising our sons and maintaining our household during my extended absence. She has sustained an island of normalcy for me in hectic times. Her faith in me, gives me the strength to aspire. Without her loving support, I would be much poorer.

Justin Gregory Nace and Joseph William Nace, my sons have demonstrated much patience. It is hard enough to be a teenager, without having a missing father. It is hard to be a father, seeing them grow almost as a stranger would. Their pride, inspires me to do my best.

Prof. Emanuel Sachs for personal and professional guidance. Our relationship has many facets. As an academic advisor, he has helped me to see both the forest and the trees. As a business negotiator, he has always bargained in good faith. As a friend, he has listened politely to all of my mid-life angst.

The Pfaltzgraff Co. for extending this sabbatical and even before that, for creating an environment where people can strive to do their best. The Pfaltzgraff company has always been an interesting place to work.

MIT for being what it is. Anyone who has known me for long, knows of my lifelong obsession with MIT. It has always represented the epitome of human intellectual endeavor, to me. The opportunity to work and learn here, is a source of great satisfaction.

Haeseong Jee and John Lee blazed the trail that I have endeavored to follow.

Professors Gossard, Boning, and Patrikalakis have all contributed their time to help guide me through this program.

The staff and students of the 3DP™ project. While they each have their own project, they share with and support the others around them. I have enjoyed their company.

In particular, I appreciate that David Brancasio has noticed, that I don't talk like other people.

Table of Contents

2. Level 2
3. Level 3

3. Design Libraries
4. Opportunities

6. Surface Texture

1. A Taxonomy of surface texture
2. Functional Texture

7. Unit Cell Geometry

1. Constructive Solid Geometry, **CSG**
2. Checkerboard
3. Abacus
4. Swiss Solid Geometry
    1. Swiss Cheese
    2. Positives and Negatives
    3. Primitives
        1. Description
        2. Motivation/Utility
    4. Disambiguation
    5. Applicability to 3DP™ Process
    6. Applicability to VLSI MCAD

8. Dimensional Tolerances

1. Trivariate Free Form Deformation
2. Utilitization for Texture Mapping
3. Separation of the Dimensions from the Geometry
4. Logical Consistancy of the Constraints

9. Mesh Regularization

1. Terminology
2. Classic Optimization Algorithms
3. Displacement Algorithms
    1. Example 1 - Relaxation
    2. Example 2 - Iterated Function System of Contractive Transformations
4. This Algorithm

10. The Implementation Specifics of Vari_4

1. Modularity
2. Module Integration
3. Module descriptions
    1. Mesh
    2. Pattern
    3. Surf
    4. Dimens
    5. Unit
4. Porting Path

11. Future

1. Alternative Surface Texturing Techniques
    1. Dithering
    2. Filters
    3. Repeated Subdivision
    4. Cellular Automatons

2. Alternative Cell Editors
3. Incorporate other Functionalities
    1. Conformal Cooling Channels
    2. Functionally Gradient Materials
4. Mesh Issues
    1. Boundary
    2. Dislocations

Index/Glossary

References

List of Figures and Tables

Figure 7.7 This shows how the small displacement of a few printing primitives, creates a void in the print bed.

Figure 7.8 This graphic representation of the displacement of primitives in 3D is how the unit cell Swiss Cheese got it's name.

Figure 7.9 This is a 2D representation of the relationships between positive, negative, and ambiguous regions.

Figure 7.10 This shows how printing primitives can be located by their centers.

Figure 7.11 The analytic geometry involved in the calculation of a tool tip point from a contact point, in 3 axis machining using a ball end mill.

Figure 7.12 This shows the unusual shape of the tool path relative to the part surface from which it is generated.

Figure 7.13 An offset surface calulated from a part surface, which is a step in the process of determining a "tool tip" surface.

Figure 7.14 This is a 3D unit cell of Swiss Solid Geometry.

Figure 7.15 This shows how print decisions are made for each print location by the Dilation Principle.

Figure 8.1 These are some of the primitive geometric objects used for Constructive Solid Geometry.

Figure 8.2 This shows how a Bezier map is used to represent a surface, from control points.

Figure 8.3 This shows how Free Form Deformation is used to shape simple objects into more comlex objects.

Figure 8.4 This demonstrates how a single unit cell is copied in order to fill a mesh.

Figure 8.5 This demonstrates how the interior of a region is mapped by FFD, the edge of the pattern does not match the boundary of the control polygon.

Figure 8.6 This shows that for an FFD of degree 1, the boundary points of the initial cell do get mapped to the boundary of the final cell.

Figure 8.7 This is a representative technical drawing showing the usage of geometric dimensioning and tolerancing.

Figure 8.8 This shows how geometric dimensions vary when they have been copied by an FFD into a mesh.

Figure 8.9 This is a conceptualization of the definitive information that is contained in a unit cell, ie the dimensions, by themselves.

Figure 9.1 This compares an irregular mesh to a regular mesh.
Figure 9.2 There are two views of the elements of a mesh, the nodes and the cells.
Figure 9.3 Each vector is a mesh node displacement, and each is a partition from the generalized n-vector, X.
Figure 9.4 This shows how a mesh can be folded if the nodes of adjacent cells are pushed past each other.
Figure 9.5 This is a representative structural problem.
Figure 9.6 These are examples of contractive linear transformations, and the orbit of an arbitrary point under their influence, as they move to a common limit point.
Figure 9.7 This is a linear affinity with the orbits of several points are marked.
Figure 9.8 This is the compact set of points that an point in the plane will move onto under the influence of an IFS. It is a Chaotic Attractor.
Figure 9.9 The displacement calculation of a node based on the correction needed for dimensional accuracy.

Table 10.1 These are the modules that plug into the main program, Vari_4
Figure 10.2 Program Structure

1. Introduction

1.1 Description of Solid Freeform Fabrication (SFF)

1.1.1 The manufacturing concept

Traditionally, industrial parts are made by material removal processes such as sawing, drilling, turning and milling . These processes are subtractive in nature. Part fabrication starts with a block of raw material, whose shape is indefinite. The desired shape of the part is achieved by a succession of machining operations. The initial operations remove relatively large amounts of material leaving behind a coarse approximation of the desired shape. Progressive operations remove less material as they work to achieve the final shape at finer levels of detail. This process works from the outside in.

The number and the complexity of this series of operations is driven by the complexity of the desired part geometry: the more complex the shape of the part to be made, the more complex the fabrication process. This translates into increased cost and increased lead time.

Also, parts fabricated this way will be largely homogenous in terms of material composition; which sets a limit on the parts functionality.

Recently, a different manufacturing concept has emerged that is additive in nature. Solid Freeform Fabrication (SFF) is a manufacturing concept, where parts are built by the incremental accumulation of material from the bottom up. The process starts with a 3D CAD representation of the part to be fabricated.

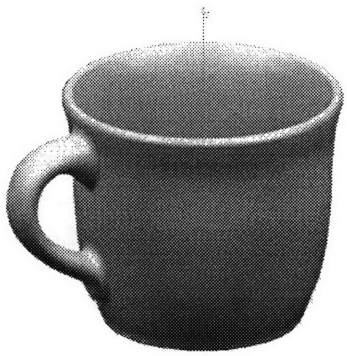


Figure 1.1 This is the CAD Representation of a part, for discussion.

Conceptually, there are three steps that are repeated to build the part. First, a cross section of the CAD model is calculated at a particular depth, Z value.

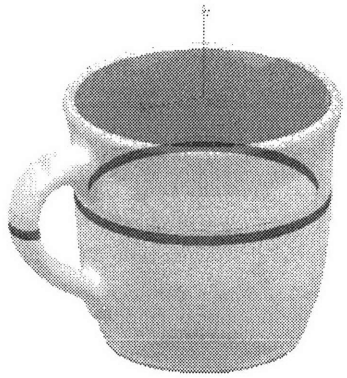


Figure 1.2 This is the computer representation of a slice of the part.

A machine utilizes the cross section information in order to construct a layer of some material, or materials.

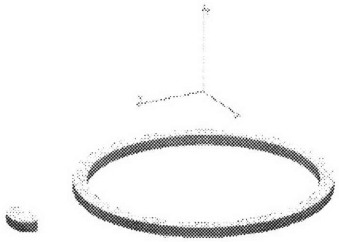


Figure 1.3 This is the slice of the part, that has been isolated for individual fabrication.

Then the machine stacks and bonds this layer onto the previously built layers.

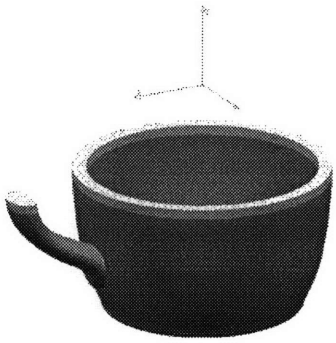


Figure 1.4 This is an idealization of how the slices are stacked after individual fabrication to build the whole part.

Physically there are many variations of this concept that utilize different types of materials, different stacking techniques, and different profiling methods.

The layering/stacking operation is the primary process that is referred to as SFF.

This process fully defines the shape and composition of the part. Generally, only a limited amount of post processing is required after the primary fabrication process. These post processes are used to improve or modify the material properties, the surface finish, or to remove other artifacts of a particular implementation of SFF.

Since each layer of the part is constructed at a fine level of detail, very complicated geometry can be built in this one process. Furthermore, since each layer is fully accessible at each step of the process; internal channels and cavities are constructed simultaneously.

The complexity of the geometry has only a minor influence on the time and expense of building the part. The time element of the build cost is basically composed of two parts; the time required to prepare a blank, fresh layer and the time required to transfer the cross section of the part into the layer. A more complicated cross section will take longer to transfer to a layer, but generally, this factor is smaller than the cost of the additional subtractive processes that would have been required to achieve the same level of detail.

1.1.2 Variations on the Theme

There are nearly as many ways to classify SFF processes as there are variations of the basic layering concept. With the exception of SLA, the processes described here are the ones that have been utilized to fabricate ceramic parts. Since 3D Systems was the first to the demonstrate SFF with SLA, no discussion would be complete without it.

1.1.2.1 - Stereo Lithography, SLA

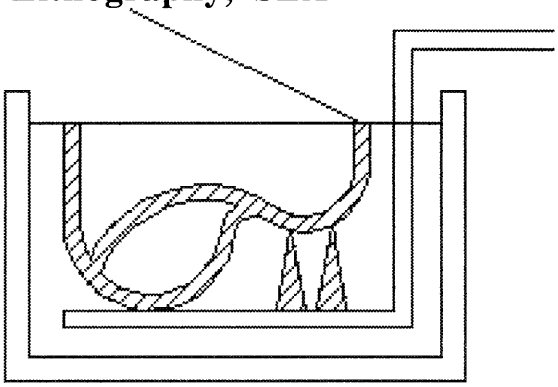


Figure 1.5 This is a representation of the SLA process.

The key element in SLA is the liquid photo-curable resin. When a laser beam traces the cross sectional information onto the surface of the resin, the resin solidifies locally; bonding to any adjacent solid material. However, if there is no adjacent solid material, then the locally solidified material may float out of location. Also, the part may be structurally unstable while it is incomplete. These conditions require that SLA parts have support structures added to the CAD model of the part, in order to mechanically stablize the part during the fabrication process. Then, in turn, the support structures require a post process to remove them.

An additional post process, is the curing of the part in a UV oven. This finishes the curing that the laser beam started.

1.1.2.2 Selective Laser Sintering, SLS

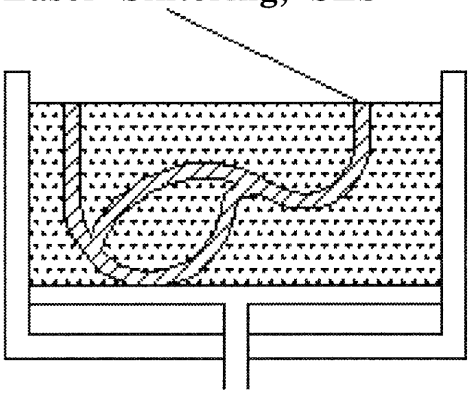


Figure 1.6 This is a representation of the SLS process.

SLS is conceptually similar to SLA. The liquid photo curable resin is replaced by a

heat fusible powder. A laser beam traces the cross section of the part on the surface of the powder bed, sintering the powder together. Aside from soft, low melting point materials; this process has been extended to a wider range of materials, by coating them with softer materials.

1.1.2.3 Laminated Object Manufacturing, LOM

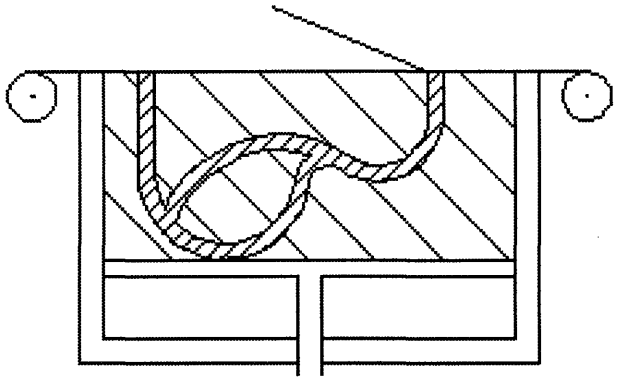


Figure 1.7 This is a representation of the LOM process.

LOM uses an adhesive backed paper. After it is fed onto the stack, a heated roller bonds it to the stack; and then a laser cuts the cross section into the new layer.

The laser also cuts the extraneous material into squares, that stack up to be cubes. This facilitates removing the part from the compact, but cavities can still be difficult to open.

LOM has been extended to ceramic materials through the utilization of tape casting.

1.1.2.4 3 The Three Dimensional Printing Process, 3DP^{TM}

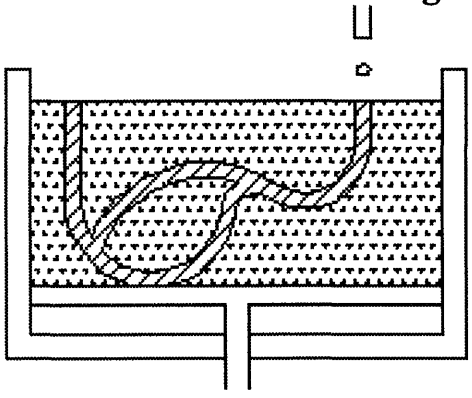


Figure 1.8 This is a graphic representation of the 3DP^{TM} Process.

The 3DP™ process utilizes a bed of powder, but the cross section of the part is created by the ink jet printing of a binder, instead of being traced by a laser. Depending on the material system of the particular application the part may require sintering, after removal from the powder bed.

1.2 The Capabilities of the 3DP™ Process

The 3DP™ printing process has many unique capabilities, the examples discussed here are not comprehensive.

1.2.1 Functionally Gradient Materials, FGM

The utility of the parts made by many of the SFF processes is limited by the availability of materials with the correct properties to work with the process. The earliest uses of SFF were to create prototypes of parts, that would ultimately be produced by other conventional methods.

The 3DP™ process, however, is applicable to a wide range of materials, since virtually anything can be produced in powder form. Furthermore, these powders do not require extensive preperation; many powdered materials can be used directly as they are supplied commercially .

The ink jet, also deposits material into the layer; providing a unique opportunity to mix materials at any point within or on the surface of the part. In addition to defining the shape of the part, the 3DP™ printing process microscopically defines the composition of the part.

Functionally Gradient Material, indicates the utilization of this concept to produce parts of heterogenous composition, that have a specific combination of physical properties that no individual material would posses on its own. A whole new category of multifunctional parts is being enabled.

1.2.2 Textured Surfaces

One characterization of the different SFF processes is based on how the part is supported during the fabrication process. The 3DP™ process is "self supporting", which means that no additional supports have to be added to the part. Any layer can possess any

number of small islands, disconnected within the layer; connected only to other layers of the part above or below.

This property of the process, combined with the fine resolution of ink jet printing; facilitates the creation of finely textured surfaces. In addition to bumps and crevices, these surfaces can even have a controlled porosity, open channels and undercuts.

Conventionally, part surfaces are textured with a variety of post processes, such as sand blasting and acid etching. Texturing is an additional subtractive process, that is limited in it's capabilities. This makes the the printing of textured surfaces, by the 3DP™ process an attractive alternative.

1.2.3 Speed

The processes that use a laser beam require the laser to trace an entire cross section of the part. The speed of this trace is limited by the amount of energy that needs to be transferred into the material, in order to solidify or to cut it. Speeding up the process requires more powerful lasers.

In the 3DP™ process it is easy to scale up the layer building process by using a printhead with multiple nozzles. A complicated cross section can be transferred into the powder bed, much more rapidly.

The variation of the 3DP™ process commercialized by The Soligen Co. uses a wide print head containing 128 nozzles, in order to build large parts in a reasonable amount of time.

The Rapid Prototyping variation of the 3DP™ process, recently commercialized by The Zcorp Co., is an order of magnitude faster than it's commercial competition.

1.3 Relationship of SFF to the Product Development Cycle

1.3.1 The Cycle

There is a strong link between SFF and the Product Development cycle. Many new products are being designed with CAD systems, which makes those products primary candidates to utilize SFF. The initial acceptance of SFF has been to build prototypes of parts and products. This reduces not only the cost of prototyping, but also the lead time of prototyping.

Some companies have begun to use these prototypes as patterns to fabricate short run, test tooling. This represents a step up along the Product Development Cycle. Another step up the cycle is being pursued by some companies, that are adapting SFF to build tooling directly. The continuing exploration of SFF is expanding its capabilities, so that SFF technologies will continue to step up the Product Design Cycle.

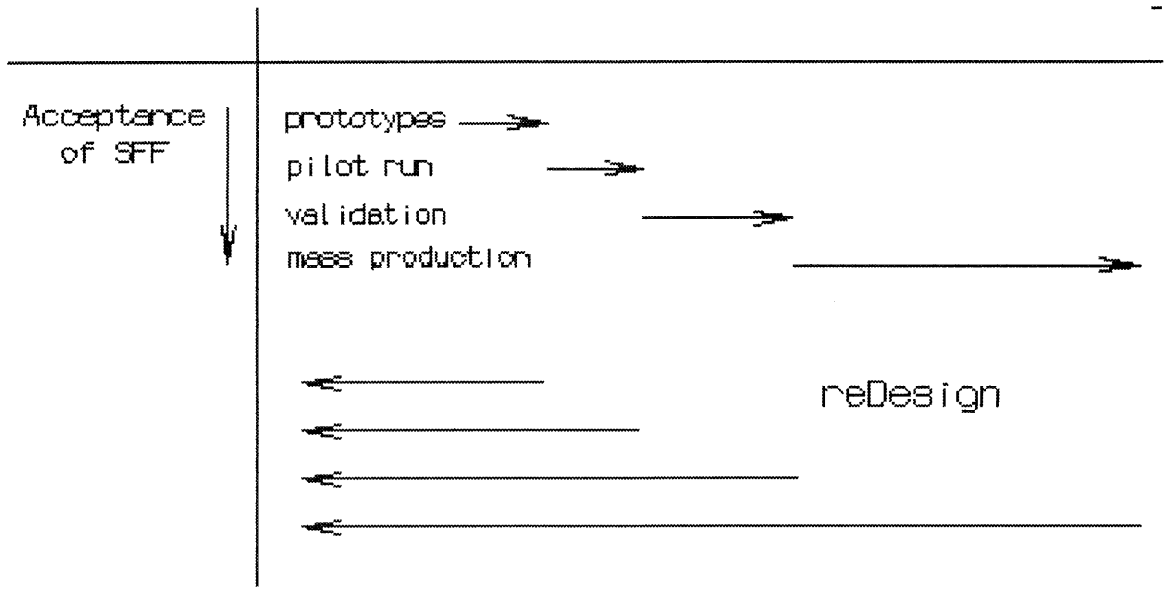


Figure 1.9 This chart shows how the commercial acceptance of SFF technologies is related to Product Development Cycle.

The Product Development Cycle as drawn in Figure 1.9 demonstrates the process when a product is scaled up to large mass production. Recent business trends are pushing companies to manufacture products in smaller batches, and to custom design products for individual customers. These trends are shortcutting the cycle, since a pilot run of a custom product, may fullfill the entire order. This situation is causing manufacturing costs to soar, due to two things; the cost of new custom tooling, and the down time due to frequent changeovers.

1.3.2 The Potential

SFF will dramatically alter the way in which products are designed and produced. There are several mechanisms by which this will happen. The first impact will be to reduce the lead time required to design new products, through the utilization of rapid prototypes.

Further gains in lead time will occur when iterations are eliminated, altogether, by the VLSI design paradigm discussed in Chapter 5.

The most profound impact of SFF is that it is a manufacturing technique that requires no tooling. Parts are fabricated directly from CAD models, the way that pages are printed by a Word Processing document. Tooling costs and tooling lead times will be dramatically reduced. Production changeovers will only consist of download time, being almost instantaneous.

This new industrial revolution is coming, Specific Surface corporation, a licensee of the 3DP process, is now shipping fully functional products manufactured by an SFF process. Their product is high temperature air filters for industrial furnaces.

2. The Organization of this Thesis

The organization of this thesis is ordinary, there is a beginning, a middle, and an end. The particular subject of this chapter is the rational for the content of those sections, a justification for that which is included and for that which is not.

The primary goal of this project is to build a Computer Aided Design system for the fabrication of fuctionally textured surfaces using Solid Freeform Fabrication technologies.

A multiplicity of minor decisions are made in the course of a project of this scope. While such decisions can be merely expedient; if they are made within a framework of incidental goals, the overall utility of any project can be greatly enhanced.

The incidental goals include:

The quality of the software implementation. This software should be easy to use and easy to understand. It should provide information and visual cues to the user as directly as possible. The level of quality should be high enough that this software could be released in some form to the SFF community. In turn, this indicates that the code should be portable to different computing platforms.

This software should be extensible. It should provide a foundation that future researchers can build upon, as they develop new SFF technologies, new 3DP™ capabilities, and new CAD paradigms.

The design process that this CAD system represents should be robust. The designer should be free to create sophisticated designs, while constrained from building unmanufacturable designs. There should be a one way flow of information from CAD to CAM, that eliminates the cycle of product development iterations.

The system should represent texture at the finest possible resolution, without limiting the range of texturable surfaces.

A list of incidental goals not only provided direction during the development of this project, but continues to tie together the various topics of this thesis.

2.1 The Beginning

Previous Work - I am not the first person to consider the creation of textured surfaces, several researchers have been involved with this idea. This chapter is meant to indicate the ideas in their projects, that have been relevant to my own project; and to give credit where credit is due.

The sections on computer coding are not meant to be critical; as a programmer, I have my own idiosyncracies. The intention of these sections is to give additional perspective to any readers who may want to examine these other codes. A cold reading of any unfamiliar computer code can be difficult.

Data Processing Background - This is a discussion that attempts to put the state of the art in data processing, for SFF, into a broader context. The goal is to build a conceptual framework that encompasses the many issues of data processing, so that new ideas can be readily positioned within the current technological dialog.

The Design Process - The VLSI design paradigm with its one way flow of information is presented and its key enablers are identified. Then two equivalent requirements are established that need to be fullfilled in order for a VLSI-like design paradigm to be implemented within the domain of mechanical part design, VLSI MCAD.

Surface texture - This section is motivational. Texture is a ubiquitous natural phenomenon, there are many ways to describe and create texture. In this section, the particular approach to surface texture, undertaken in this project, is developed.

2.2 The Middle

This section contains my results. Pricipally, that a VLSI MCAD is feasible. One requirement for such a design methodology is fullfilled by the Unit Cell type refered to as Swiss Solid Geometry, SSG. The other requirement is met by the computational algorithm described in Mesh Regularization.

While a key idea is developed in Dimensional Tolerances, Unit Cell Geometry is presented first because it preceded that development historically. It was during the evaluation of unit cell geometries that the key idea of managing dimensional tolerances,

independant from their associated geometry, emerged.

Unit Cell Geometry - This section is an evaluation of CAD modeling stlyes that can be applied to create a feature level description of texture. It concludes with a description of Swiss Solid Geometry, a unit cell that fullfills one of the requirements of VLSI MCAD.

Dimensional Tolerances - This is a discussion of geometric dimensioning and tolerancing. Dimensional relationships are central to the process of ensuring manufacturability. Recognizing the manner in which these relationships represent information, is the first step towards enabling VLSI MCAD.

Mesh Regularization - The idea developed in Dimensional Tolerances is demonstrated by the creation of a mesh regularization algorithm that is driven solely by the multiple instantiations of the Dimensional Tolerances throughout the mesh. This algorithm satisfies the other requirement of VLSI MCAD

Implementation Specifics - A key deliverable of this project is the software that implements the ideas developed in this thesis. In order to facilitate that delivery, this section contains additional documentation that does not fit well into the body of the computer code. Software documentation is a complex issue. In my way of thinking, it would be ideal if the source code and the thesis could be combined into one document, by an editor that mixes text and graphics, and also computer code that is operational. This is the "tangle and weave" concept of computer program editing. Obviously, that is beyond the scope of this project.

2.3 The End

The Future - This is a discussion of alternative approaches to surface texturing that are more stochastic than, and less regular than, the approach used in this project. Some interesting meshing issues are also presented.

Glossary - The glossary should facilitate this acronym laden discussion. It also serves to establish specific meanings for words and phrases that could have a variety of

meanings. For example, the word "cell" should not occur without one of the adjectives "mesh" or "unit".

This is a very rich field of intellectual endeavor. Nearly everyone that I discuss this subject with, immediately has ideas on the topics of Rapid Prototyping and/or surface texturing.

Several fortuitous coincidences have greatly enhanced my pleasure in this work. The close relationship between the 3DP™ process and traditional computer graphic techniques; and the relationship between the commercial acceptance of SFF and the new product introduction process. Continued incremental improvements in SFF will lead directly to continued improvements in product life cycles.

3. Previous Work

3.1 Alain Cordeau

Alain demonstrated the practical application of textured surfaces to orthopaedic implants. He developed parameters for pore size, for post processing, and for metal casting; appropriate to the manufacture of cobalt chrome steel bone implants by metal casting into a 3DP™ printed ceramic casting shell.

Alain printed his textures using a "spreadsheet" approach to the data issues. He meticulously hand coded the printing transitions needed by the 3DP™ process for a single cell on each layer, and then used spreadsheet software to replicate the instructions. This limited him to certain regular surfaces, such as planes and cylinders.

3.2 Haeseong Jee

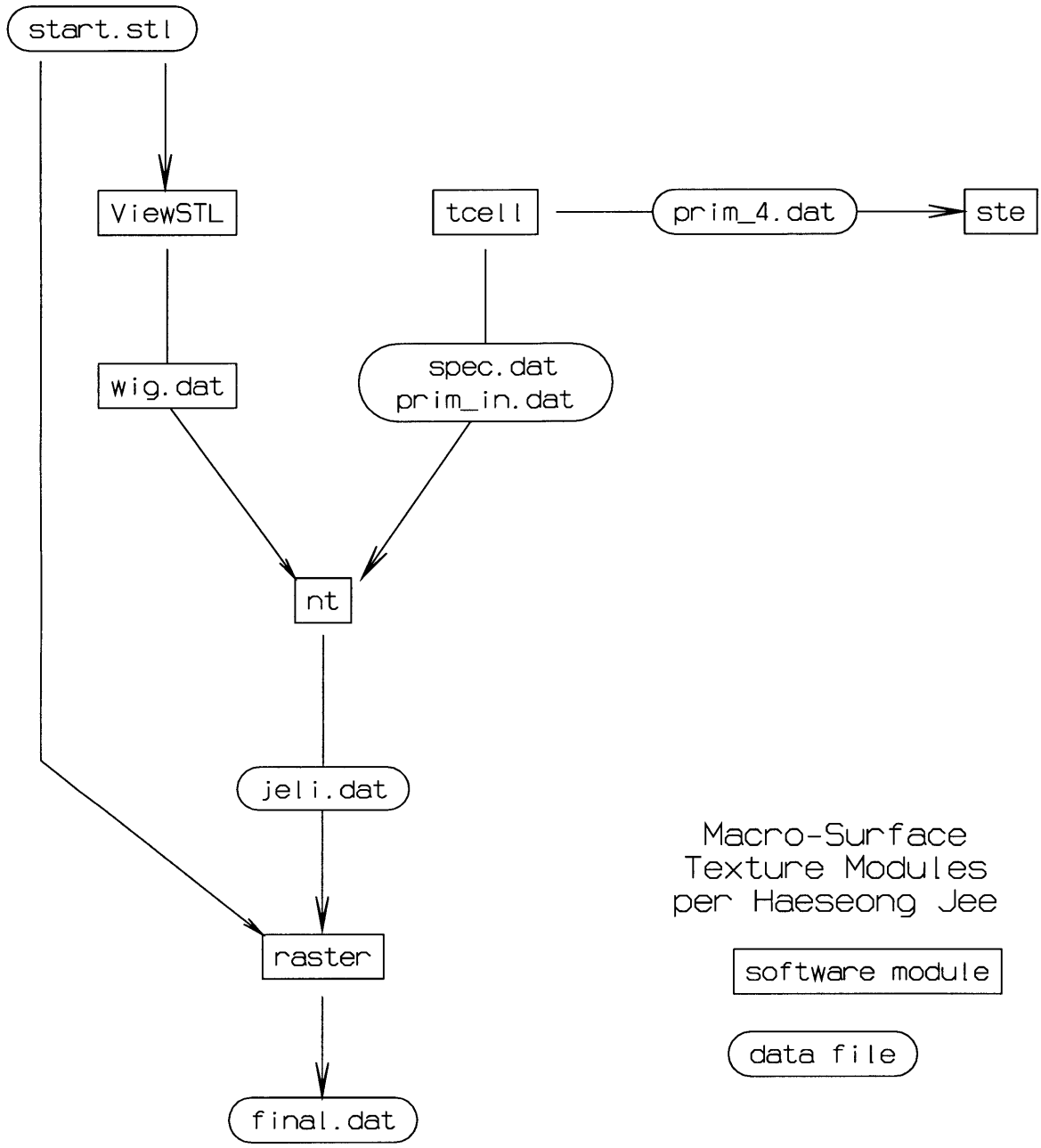
3.2.1 Texturing Process

I worked with Haeseong during his last semester at MIT. His work is the immediate predecessor to this work.

Haeseong developed several tools for the creation and manipulation of surface texture with the specific objective of linking manufacturing constraints into the design process. His tool kit demonstrates the basic concept that texture can be managed by Computer Aided Design during the design process. Haeseong created 3D CAD programs in C, using the IRIS GL library. His user interface consists of a combination of GL menuing and command line input.

He built a CAD process consisting of a pipeline of computer programs and computer files. See figure 3.1 on the next page.

The part that is to be dressed with texture is referred to as "start.stl". ViewSTL is used to reapproximate a portion of the stl file as a nurb surface. This is a 3D interactive process that requires the user to create the control points of the nurbsurface, by using the mouse to point to the displayed stl object. The user must indicate locations in a row-column format, where the number of rows and columns matches the order of the intended surface approximation. When the process is finished, ViewSTL writes out a file called "wig.dat", that contains the X,Y,Z values of the control points.


start.stl
ViewSTL
tcell
prim_4.dat
ste
wig.dat
spec.dat
prim_in.dat
nt
jeli.dat
raster
final.dat
Macro-Surface
Texture Modules
per Haeseong Jee
software module
data file

tcell is an interactive 3D tool to design a single unit cell of texture. The unit cell consists of a collection of regular quadrilateral "voxels". tcell writes several files when the unit cell design process is complete. "spec.dat" and "prim_in.dat" describe the unit cell. "prim_4.dat" is a file that contains a side by side arrangement of four copies of the unit cell.

ste is an interactive 3D program that renders the group of unit cells defined by "prim_4.dat". Its purpose is to aid the designer, in regards to the manufacturability of the unit cell; by providing visual feedback of the cell "as printed". ste simulates the 3DP™ printing process and displays the unit cell with spheres and cylinders representing the droplets and passes of the process. The user can rotate the unit cells around, while the orientation of the print bed remains static, in order to view the "as printed" cells, as they may be printed at various orientations in the powder bed.

nt brings together the nurb approximation of the part from "wig.dat" and the unit cell from "prim_in.dat" and "spec.dat"; by building a mesh on the surface, and then copying the unit cell into the mesh cells, using a trivariate free form deformation. nt writes "jeli.dat", which is the vertex information for the voxel primitives, as they are copied in the mesh cells.

raster calculates the printing instructions, for the 3DP™ process, necessary to print the part as it is described in "start.stl", and the texture as it is described in "jeli.dat".

3.2.2 Discussion

Since the quality of the mesh controls the quality of the copies of the unit cell, Haeseong uses an Energy Minimumization algorithm to produce a regularized mesh, as orthogonal as possible given the constraint of lying on the surface. He associated an elastic potential energy to the mesh, by conceptually connecting the nodes of the mesh with springs. As the energy of the network of springs is minimized the shape of the cells is regularized.

The droplets of binder that the 3DP™ process ink jets, create discrete primitive features, usually represented by circles, on the surface of the powder bed. Haeseong, then checks manufacturability by examining the copies of the cells in the mesh to see if each of the mapped voxels can contain at least one of these primitive features. He outlines a procedure to ensure manufacturability, that specifies that the entire mesh should grow in size, in order for each mesh cell to meet this criteria.

Global regularity means that the mesh will grow in order to accommodate the most constrained mesh cell.

3.2.3 Style

Haeseong's programs are monolithic, not modular. He created many very similar versions of the same basic program, by saving his work frequently with new names. Unfortunately, I could not discern a naming convention. I picked one version of each to study, but they may not have been his final, best effort.

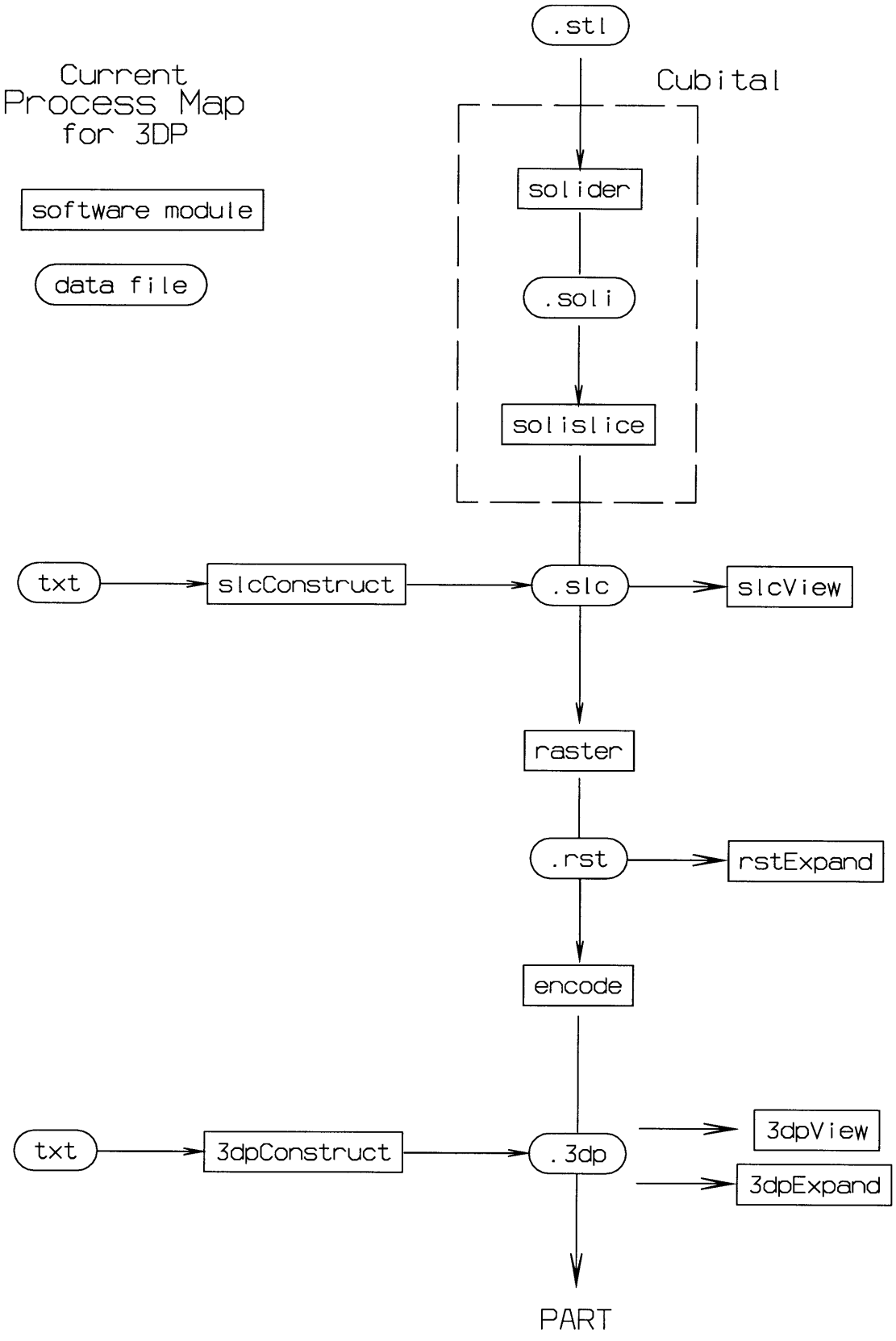
3.3 Sang-Joon John Lee

3.3.1 Printing Process

John developed the software that is currently in operation to process parts into a format, for the 3DP™ process. He created a pipeline of sequential programs that progressively converts an STL file into the correct format for the 3DP™ process. See figure 3.2 on the next page. He also developed debugging tools for each stage of the translation. John wrote his programs in C, using the Motif toolkit for menuing, and using the Motif DrawingArea widget for drawing. These programs are inherently two dimensional.

The process starts with an STL file. The 3DP™ consortium has worked on projects from many industrial sources, and the STL file has been used as the standard. Unfortunately, the STL representation of a 3D CAD object is neither accurate, nor robust, see Chapter 4. John had to put a lot of effort into compensating for bad STL files, hence the need to examine the part files at each stage of the process.

In fact, before translation begins the STL file is verified and properly positioned in the print bed using commercial software from Cubital. Solider has the capability to make some repairs to a bad STL file. If the STL file is usable, another Cubital product, solislice, is used to slice the 3D STL object into layers of 2D polygons. The layer information is stored in an SLC file.


Current
Process Map
for 3DP
software module
data file
.stl
Cubital
solider
.soli
solislice
txt
slcConstruct
.slc
slcView
raster
.rst
rstExpand
encode
txt
3dpConstruct
.3dp
3dpView
3dpExpand
PART

Next, the 2D polygons are rasterized. In this process, rasterization consists of replacing the 2D polygons, on each layer with scan lines across the polygons. Each scan line represents the passage of an individual ink jet over the print bed. For each pass, the location of the pass, and the location of the ON/OFF print commands along each pass are written into an RST file.

Then, the raster file is encoded. The passes are associated with particular jets in the printhead. The boundary segments are traced by deflection of the ink jet droplets, and the interior regions are filled with print patterns. The 3DP™ printing process is not fixed to a rigid raster; the droplets can be deflected along the slow axis, to improve the quality of edge tracing; this technique is refered to as "Proportional Deflection". Other special printing styles such as "Exits Only" are specified in the encoding process. Lastly, all of this information is written into the a binary file format, that the alpha machine reads, as printing instructions.

3.3.2 Other Tools

The debugging tools, slcView and 3dpView are used to graphically visualize their respective part files. rstExpand and 3dpExpand display specific numeric information contained in the part files, in human readable form.

The slcConstruct and 3dpConstruct, which where originally used to debug the file processing pipeline; remain in use to build specialized test parts for student projects. Each of these programs accepts a text file that is properly formatted, and directly writes a binary file of the appropriate format.

John also created several programs to modify SLC files; such as slcOffset which compensates the 2D polygons for the size of the print primitives, and slcRepair which allows the user to interactively modify the vertices and the orientations of individual polygons.

3.3.3 Style

John's code is highly organized. He used a motif like naming convention for his functions and parameters. He likes to store data in lists. There are very few bugs, I was

surprised to find any. John had to concern himself with portability between a DEC workstation and an SGI workstation that utilize different byte orders. His utilities, list, byte, file handling, etc. are in modules; but he did not use makefiles.

John installed his code on Pesta, as the operational tools. He was also the administrator of Pesta, so he was selective about which versions he left behind.

4. Data Processing Background

This chapter does not contain a description of how to create a file, for the 3DP™ process, that information is included in 3.3 Sang-Joon John Lee.

Design and business opportunities relating to improved data formats are discussed in Chapter 5 The Design Process.

4.1 The role of CADCAM in SFF

CADCAM is the technology that enables SFF. Not only does an SFF process start with a 3D CAD model of a part, but CAM is the only reasonable way to provide the large amounts of data required by the process. A file of operation codes needed to control an SFF machine can easily reach 100 megabytes.

In manufacturing, there is a large number of disparate processes by which a part can be fabricated. Additionally, there is a multiplicity of machines of different make and model available for each process. This level of complexity is managed by an operational separation between CAD systems and CAM systems. CAD systems are used to create a model of the part; in particular, the geometry of the part. CAD systems are also used to document the non-geometric properties of a part, via markups and part lists. This is the extent of the design process.

CAM systems take the CAD model, and create the **NC** instructions required to operate a specific manufacturing machine. Such a program, also allows the visualization of the manufacturing process, and reports areas of the part that can not be fabricated with a particular machining operation. This information is used for process planning, so that another machining operation can be applied to finish the part.

Sometimes, the design is deemed problematic. In this situation the part needs to be redesigned in order to be manufacturable. Iterations are inherent in this process.

Surface quality and composition are part properties that are not represented geometrically. Generally, surface quality is marked on a drawing of the part, and composition is indicated in the bill of materials. Each representation of a part property is meant to facilitate communication in the most useful form, the most direct as to the manufacturing process. Surfaces are treated as geometric ideals, slick, because there has

been no way to manufacture texture under numerical control. Likewise, there has been no way to numerically control the microstructure of a part.

With the emergence of Solid Freeform Fabrication, it is now possible to numerically control surface quality and composition. However, in order to utilize these new capabilities, new CAD and CAM tools are needed. There are two aspects to this need; fundamentally, there needs to be a way to represent these features in a CAD system, geometrically; and there needs to be a way to exchange this information between CAD and CAM systems.

4.2 Relationship of Visualization to Manufacturing

4.2.1 Pipelines

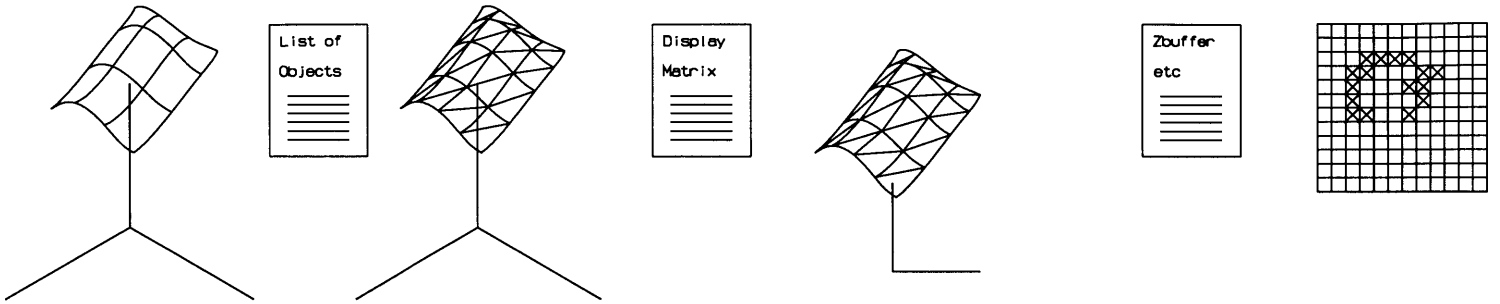
While a CAD system represents the geometry of a part with a few floating point numbers, it displays the part with a large number of integer values. The information has to be converted from a very compact form, to a form that represents instructions to the display device. Computer graphics is a well developed field, and a specialized method has been refined to perform this process. See the chart, figure 4.1, on the next page.

Starting with a surface or solid representation of the part, the part is approximated as a collection of planar facets, tessellated. An object list determines which surfaces and solids are to be displayed. Then each facet is projected onto a 2D plane, that represents a view of the part. The parameters of the projection are contained in a display matrix. Rasterization, refers to the numerical approximation of the floating point values used so far, into integer values. A zbuffer keeps track of which facet is closest to the view plane, when it is rasterized, so that the correct 3D logic is maintained.

The display pipeline is implemented for speed; this is a large task, that has to happen quickly; so the algorithms have been optimized and specialized hardware has been developed. Many of the operations are performed in parallel, so that the exact execution does not proceed in whole large pieces.

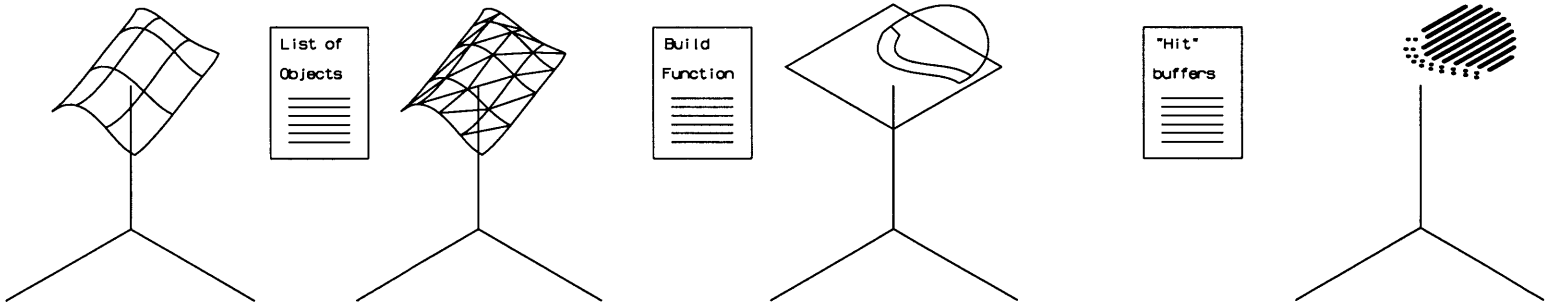
Graphic Pipeline

CAD Model Tesselation Projection Rasterization



Build Pipeline

CAD Model Tesselation Intersection Discretization



The "Build Pipeline" by which CAD models are converted to SFF machine formats is entirely analogous to the display pipeline. This is especially true for the 3DP™ process, because of the discrete nature of ink jet printing. For each build, a number of parts are placed into the printbed, with some orientation; that determines the cut plane. The principal difference is that the 3D facets are intersected by a cut plane, instead of projected onto a view plane. Another difference is that there is a stack of cut planes instead of one view plane.

A "hit buffer" can be used to count the exact number of particular types of droplets, as instructions to print them are generated. This would be useful for medical applications, that need precise compositions; or statistically driven algorithms, that control such things as color printing. Currently, there is no "hit buffer", but it is just one of the many ideas that are suggested by the build pipeline analogy.

4.2.2 Feature Implementation

	native CAD	.stl	.soli	.slc	.rst	.3dp
	files	collection of	collection of	layers of	layers of	printing pass
		3D triangles	3D polygons	2D polygons	raster lines	transitions
Exits Only					X	X
Macro Drops						X
Texture		X	X			X
Color						X

Table 4.2 This table indicates the location in the build pipeline where different process capabilites are best implementationed.

Whether it is explicitly implemented or not, this conceptual pipeline is a convenient tool to discuss issues of the software capability.

The position in the pipeline at which a feature is implemented affects its capabilities. For any particular range of the pipeline, only certain data objects are available for modification. Aside from the basic translation from one data type to another, the pipeline also allows additional process information to be incorporated at each stage. For these reasons, it is more appropriate to implement some features at one place than at another. Many of the process planning improvements implemented by John Lee, are implemented in the .slc, .rst, and .3dp formats, low level processes.

Texture, however, is a fully 3D issue; and needs to be dealt with at a higher level,

in a CAD, .stl, or .soli format. In this project, texture is handled at the stl format level. Vari_4 reads an stl file; and then writes a modified stl file.

4.3 Current Situation

The current situation consists of a collection of heterogeneous tools, developed one at a time to satisfy one specific need. While prototypes are being fabricated rapidly, there is not enough consolidation of effort and awareness of larger issues; to produce an industrial metaphor of CADCAM for SFF.

The National Science Foundation is supporting the search for more robust formats. It hopes to see a Solid Interchange Format, SIF, and a Layered SIF, LSIF, format developed. These hypothetical formats are intended to solve two issues: representation and exchange. Representation is the ability to describe the new capabilities of SFF. Exchange is the ability to communicate more information about the part, in a more robust fashion.

4.3.1 File Formats

The development of file format standards is lagging far behind the development of new SFF technologies. Since 1989, when 3D Systems first made the STL file format available, very little has changed. The STL format is a simple list of 3D triangles, it does not contain any topological information regarding the structure of the list. There is no way to verify the accuracy of the file. Files in this format are frequently corrupt, and unusable without "repair". STL files for complex parts are large, which makes them difficult to exchange and to process. None the less, the simplicity of the format and the lack of alternatives, has led to the widespread adoption of the format by the SFF community.

The current CAD exchange, IGES, is not capable of satisfying the conditions of representation and exchange, in other than a trivial extension. The emerging standard, STEP, may be extensible to the needs of SFF, but that work will not be available in the near future.

4.3.2 Vendor Supplied Software

There is no standardization of the CAM software. Each manufacturer of SFF equipment supplies their own software to operate their machine. Typically, they accept an STL file, and slice it; then they convert it into lowlevel instructions to operate their

machine. At this level they implement process specific capabilities, such as 3D Systems, Starweave. Starweave fills solid regions with a particular pattern that reduces the amount of work that the laser beam needs to do, yet still produces a stable part.

Even though the many SFF processes are very different at the lowest level, there are many similar processing steps; that could be captured in a standardized format. Examples of this would be Automatically Programmable Tool, APT which captures the lowlevel movements of a machine tool in a generic format; that many different machine tools can utilize. Another example are page description formats such as Postscript, which describes a page to a printer, in a format that many printers can interpret.

4.3.3 Third Party Software

While, some CAD suppliers have incorporated STL output into their programs, many have not. This has allowed third party companies such as Imageware and Deskartes to enter the market with softwares that convert CAD models in a variety of different formats into STL format. While these softwares are at the best position along the build pipeline to establish SIF type formats, there is little incentive for third party vendors to add features to a CAD model.

5. The Design Process

5.1 VLSI Paradigm

The Computer Aided Design of VLSI circuits is an information intensive process, the behavior of tens of thousands of individual circuit elements has to be specified. Then these circuits have to be accurately fabricated in order to create a functional device. In spite of this situation, ECAD techniques have been developed that guarantee that a functional device can be built on the first pass, without iteration.

Some of this capability comes from the large flexibility of modern electronic part fabrication technologies; however, a lot of this capability comes from the VLSI design pardigm.

Several concepts in the electronic part design process make this possible. Principally, the process from CAD to CAM is broken into separate conceptual levels of abstraction. Abstract design components, are defined at each level, whose behavior is appropriate to that level. Furthermore, this behavior is constrained so that the component behaves correctly at lower levels of abstraction. For example, transistors are a primitive design component at one level of abstraction; but they are fabricated as a sophisticated geometric combination of materials, at a lower level.

5.2 Mechanical Design Abstraction

5.2.1 Level 1

CAD techniques for the utilization of SFF, and the 3DP™ process can be developed to several levels of sophistication. The initial work in this field has been to enable the basic functionality of Rapid Prototyping. These tools are described in Chapter 4. From the point of view of current commercial CAD systems, SFF technology has been treated as an alternative output, via the writing of an STL representation of existing CAD data. At the other end of the spectrum, is the work being done to improve the machine control pertinent to specific SFF technologies. These approaches preserve the status quo.

5.2.2 Level 2

The next level of development is represented by CAM software. In Numerically Controlled machining specialized software packages are used to process a part down the

path from design to manufacture. While both levels create NC instructions, the distinction of this level from the previous level, is the amount of information that is added to the part description. Functional Features that may have been annotations to the part in the CAD system are instantiated in terms of specific machining operations and the "additional" geometry that supports such operations. This extra geometry is "additional" from the point of view of the designer, but it is essential to the detailer.

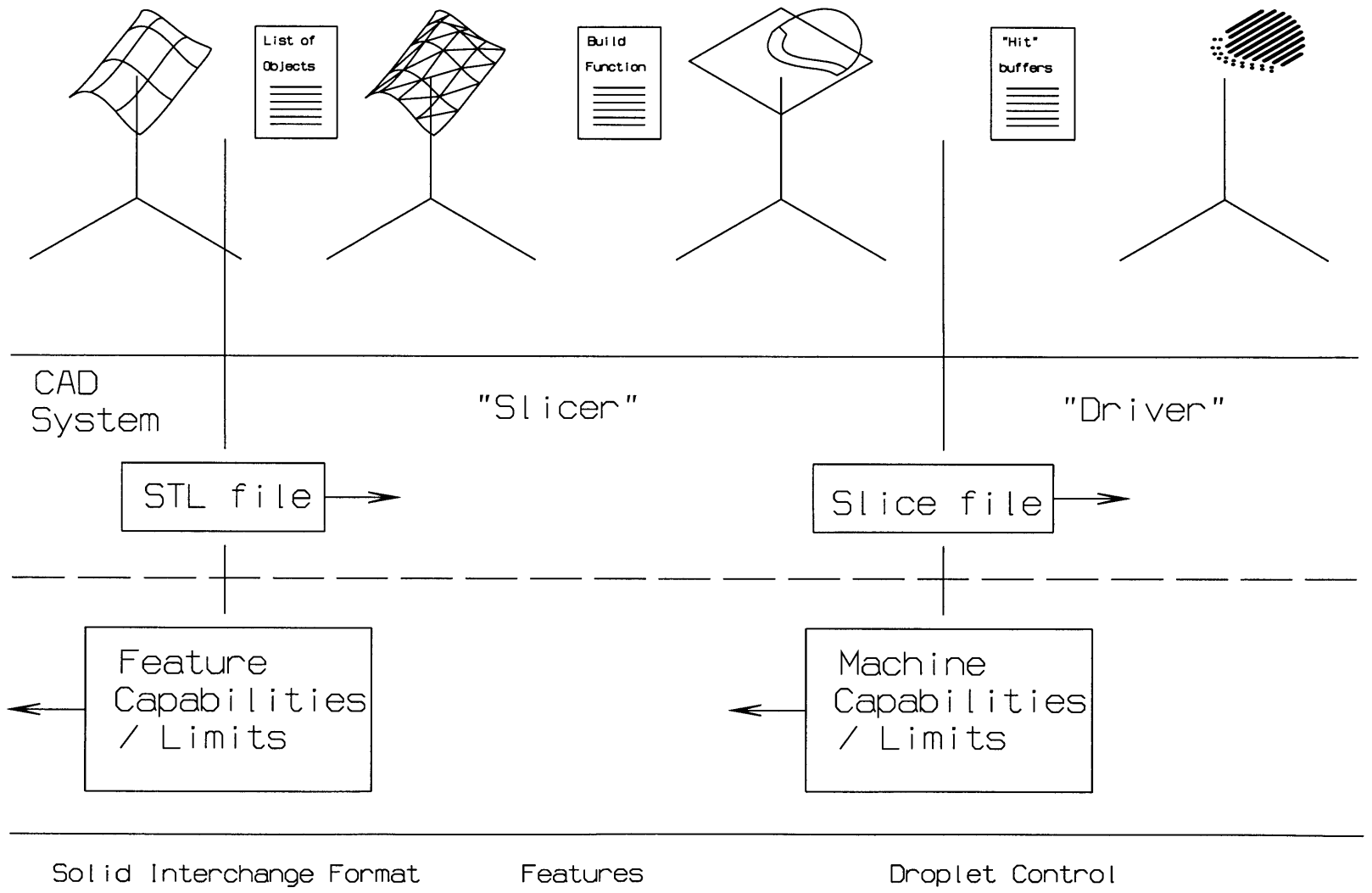
A CAM software for SFF should accept the geometry of a part from another CAD software, then detail the part with additional geometric information, and then create machine operations for the fabrication of the part. This process is the mechanism that enables the utilization of the advanced capabilities that SFF offers.

This is the level of SFF machine vendors and third party software suppliers.

Figure 5.1 shows this level of abstraction on the build pipeline. The chart shows how the data flows from design to fabrication. "Driver" is a low level software, that provides an Applications Programmable Interface, API. The chart also shows how information must be conveyed up level, in order for the previous processings to be accurate. The issues of Solid Interchange Format, Features, and Droplet Control are positioned logically across the bottom of the chart. The boundaries are not precise different implementers such as SFF machine vendors, and third party software developers position them differently.

The data format issue discussed in Chapter 4 of representation can be handled internally by the CAM software; and the issue of exchange can be ignored; for input, by the utilization of existing standard formats, such as STL; and for ouput, by considering it to be machine specific, or even proprietary, from the point of view of machine vendors.

This is the level that is implemented in this project. A higher level of implementation is beyond the scope of this project.


List of
Objects
Build
Function
"Hit"
buffers
CAD
System
"Slicer"
"Driver"
STL file
Slice file
Feature
Capabilities
/ Limits
Machine
Capabilities
/ Limits
Solid Interchange Format
Features
Droplet Control

5.2.3 Level 3

There is, however, another level of development that mechanical CAD systems should evolve to, and that is the level suggested by the VLSI paradigm.

Among the many benefits of SFF is its wide flexibility. In particular, the 3DP™ process, with its capability to control composition at microscopic levels, is a fabrication process of unprecedented flexibility. This flexibility is one of the two key enablers of the VLSI design paradigm.

The other enabler of the VLSI paradigm is the identification of levels of abstraction for the design process. While this is not a trivial problem, the microstructural capabilities of the 3DP™ SFF process by which parts can be fabricated with the specific control of individual functional features, allows the problem domain of part design to be separable. Prior to the development of these capabilities, all of the physical properties of a part, such as weight, strength, and heat capacity; were determined entirely by just two parameters, the part geometry and the bulk composition. In this situation the physical properties of the part are codependant on each other, so that the problem of designing a part with specific properties is complex, and is based on "trade offs" between the properties. This separability of the problem domain has been demonstrated by Xiarong Xu, in his work on the design of conformal cooling channel functional features.

For the topic of functional features such as surface textures, where composition is not currently being modified, the situation can be described as follows:

As noted in the discussion above on NC CAM, there is geometry, and then there is geometry. In order to have multifunctional parts designed at a high level of abstraction it is necessary to develope geometric models of functional features that have two modes of behavior. The higher mode, for conceptual design, is simpler; it is managed with a relatively small number of parameters, and it is represented by a subset of the complete description. The lower level mode is a full description of the feature, sufficient to fabricate the feature.

In order to be a successful abstraction, there are two criteria:

1. That the parametric information of the holistic, complete model of the feature be separable into logically distinct subsets.

2. That any processing or calculation performed on or with one subset is consistent

with the parameters of the other. This consistancy is a generalization of the concepts of Design Rules, and of Manufacturing Constraints.

This is similar to the facility that many CAD systems have, to design a family of parts, using parametric geometry. The geometry of some part, such as a screw, is stored with parametric dimensions for such featuress as heighth, width, and thread pitch. When a user wants to define a specific screw, they assign values to the parameters that meet their concurrently desired design criteria. There are two differences; first is the conceptual range of the parameters, they are typically established for the gross geometric features of a part, not for microstructure, and not for composition. The second difference is that the parameters are not constrained by, nor carry with them, the manufacturing constraints required to fabricate the feature.

5.3 Design Libraries

There is a new discipline implicit in this discussion. These abstract descriptions need to be created and maintained. In order to create these descriptions there needs to be robust modeling tools that satisfy the two criteria given in the previous section. In order, to maintain, to organize, and to make available to designers; libraries of these models of functional features need to be established.

5.4 Opportunities

When such models of functional features are defined, it will be possible to design and to fabricate parts utilizing a one way flow of information similar to VLSI. This will eliminate the iterative nature of mechanical design.

The NSF has even proposed that generic mechanical part manufacturers could be established, since the tight coupling of design for manufacturing would no longer exist. It would be possible to fabricate functionally equivalent parts, while utilizing totally different SFF processes, operated by totally distinct concerns.

It will take some time for industry to adjust to these concepts. There may be some professional resistance to the introduction of ambiguity, that working at a high level of abstraction introduces. Traditionally, the designers of mechanical parts have been intimately involved with the manufacturing methods, used to fabricate their designs.

6. Surface Texture

6.1 A Taxonomy of Texture

Physical surfaces, whether natural or man made, differ from the CAD idealization of them. A taxonomy has been developed for the descriptions of these surfaces, based on two key parameters. [RAV nn] First there is the measure of the uniformity versus the randomness of the texture, which forms a scale with three primary categories:

Roughness refers to the situation where the distribution of the surface features is fully randomized.

Texture refers to the situation where the distribution of the surface features is partially randomized, like swirls on a stucco wall.

Pattern refers to the situation where the distribution of the surface features is completely regular, like a checkerboard.

The other parameter is the scale of the texture. At some scale, every physical surface is composed of discrete particles; and at the other end of the scale, every surface has some finite extent. The scale of the texture for a surface lies within that range. At the lower end of the range, the surface features that constitute the texture can be the discrete particles of the physical surface. At the upper end of the range, texture is not an appropriate description for a surface feature whose scale approaches the over all size of the surface.

A complete description of a particular texture requires that two things be specified, the surface feature that comprises the primitive unit of texture, and the manner in which that primitive unit is distributed on the surface.

In this taxonomy, the textures created in this project would be more accurately refered to as patterns. The word, texture, however, is used broadly in computer graphics literature to describe a broad range of surfaces; as it will be here.

6.2 Functional Textures

In mechanical part design, the surface quality of the finished part contributes to the functionality of the part. While smooth surfaces are useful for reducing friction between moving parts, other surfaces can contribute other functionalities. Javier Banos has

demonstrated that the texture of the surfaces of heating and cooling channels enhances the heat transfer between the working fluid and the mass of the part. Alain Cordeau has demonstrated that a surface with a controlled porosity enhances the boney ingrowth of orthpaedic implants.

The consideration of functionality, imposes additional constraints on the methods of design and fabrication of particular surface textures. Principally, the geometric dimensions of the texture primitives must be controlled. The size of negative features, voids such as pores, is as important as the size of the positive features.

Additionally, the topological issues of connectedness and completeness (the lack of flaws), are important in the formation of functional features, such as a network of capillaries.

These considerations, lead to the specific methodology for the design of surface texture, utilized in this project. A regular grid, a mesh, is used to position the features on the surface; so that the features are uniform in size, regularly spaced, and nonoverlapping, yet adjacent. Furthermore, the grid will be composed of quadrilateral elements, not triangular elements. This insures a correct orientation of adjacent cells, so that the topological connectivity of the cells, can be maintained. This also facilitates the design process, designing a unit cell of texture, and anticipating the complete texture, is more natural with four sided domains, than it is with three sided domains.

7. Unit Cells

In order to model texture as a repitition of primitive elements, a model of an individual unit cell is required.

Many different unit cell concepts were considered in the course of this project. There are many conflicting goals to consider when choosing a model for the unit cell. One consideration is the applicablity of a unit cell to differing surface regions. There are non-trivial differences between planar regions and highly curved free form surfaces. Even amoung planar regions, differences in orientation in the print space, are very complex.

Another important consideration is the resolution of the cell. The more closely that geometric cell primitives resemble process primitives, the more difficult the cell is to map onto a surface without error. This problem is further exacerbated by the level of abstraction that is intended. A generic cell model that would serve the SFF community, will not take advantage of the finest resolutions of which the 3DP™ process is capable. A model that closely mimics the 3DP™ process will be too specific for widespread SFF usage.

In terms of the 3DP™ process, the most dificult element to manage is the contiguity and noncontiguity of printed features. When a unit cell is mapped, and then discretized; it is difficult to maintain the topological connectedness of primitives that are meant to be connected, and to maintain open pores where they are intended.

In Chapter 8. Dimensional Tolerances, the idea of separating the dimensional information from the geometric information, for separate processing is presented. Some of these unit cells are readily adaptable to such an approach, some are not.

7.1. Constructive Solid Geometry

CSG is the approach to solid modeling that many full featured CAD systems utilize, to represent complex solid objects. A complete system consists of primitive elements such as cubes, spheres, cylinders, and cones; together with operations to combine these primitives into more complex objects such as unions and intersections.

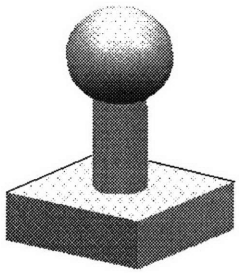


Figure 7.1 This is a complex object contructed from simpler geometric primitives.

One thing that has to be considered with this unit cell, is the integrity of the geometric forms. There are two possible approaches. The CSG primitives can be treated as exact surface definitions in the unit cell, and then allowed to take on whatever form the mapping gives them as they are replicated across the surface region. This is easy to implement, but the elements of the textured surface could be quite odd.

The alternative is to map the locations and spatial relationships of the CSG primitives onto the surface; and then instantiate exact geometric objects there. The size parameters of these objects would be the nominal values of the objects as they are defined in the unit cell. This method motivates the separation of the geometric information from the dimensional information.

While this process captures the design intent, it is difficult to visualize the ultimate variation in the texture that will occur; during the design process. In this situation, the CSG primitives will stay the same size and shape that they started out with.

This unit cell is abstract enough to be useful to the SFF community.

Haeseong implemented a subset of CSG modeling as his unit cell model, using the first approach above. He used regular quadrilaterals, that he refers to as voxels. In order to insure manufacturability, and to constrain the design process, he developed relationships between voxels and 3DP™ process primitives. The minimum size of each voxel is constrained so as to be able to contain a printing primitive regardless of the orientation of the voxel. He used a discrete placement scheme, dimensional stoppers, to control voids, negative spaces.

7.2 Checkerboard

A unit cell editor implemented in this project is the checkerboard. It was inspired by Alain Cordeau's graphical short hand to describe patterns.

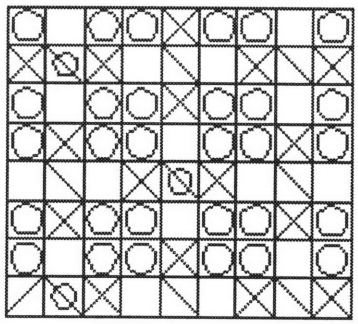


Figure 7.2 This is an example of the method that Alain Cordeau used to represent texture patterns.

The different symbols in the pattern not only indicate different heights, some of them indicate the occurance of occluded voids.

The checkerboard accomplishes the same effect, by indicating the presence or absence of a voxel, at different height levels. As such, the checkerboard is a 3D binary object, which is easy to implement. The checkerboard is conceptually very similar to the cell that Haeseong implemented, however its manipulation is quite different.

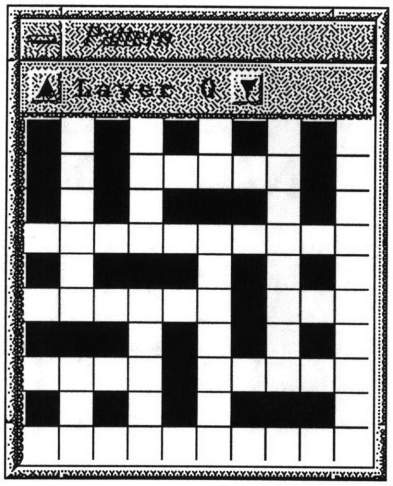


Figure 7.3 This is how the checkerboard unit cell is displayed.

The difficulty with the checkerboard is that the size of the voxels is uniform, while the minimum sizes of positive features and the minimum sizes of negative features, for the

3DP™ process, are different. The checkerboard needs to be constrained to represent the smaller of the features, and to build the larger from finite collections of voxels. The checkerboard is implemented so as to join adajcent regions, insuring the contiguity of positive regions. Dimensions can be associated with collections of voxels, in order to capture the design intent of the particular arrangement of individual voxels.

7.3 Abacus

Abacus is a unit cell concept that is specific to the 3DP™ process. It is an ambitious concept to control the low level printing of individual process primitives. The idea is inspired by pixel editors in paint programs, and adapted to represent the proportional deflection capabilities of the 3DP™ process. Proportional Deflection is the ability to deflect a droplet in a direction normal to the direction of the movement of the print head.

As explained in Section 3.3.1, the data for the 3DP™ process is organized, first by layers that represent Z values, typically spaced 170 microns apart. Then it is organized by raster lines, that represent ink jet passes. The location of these raster lines represent Y values, typically spaced 170 microns apart. Then the spacing of the individual ink jet transitions, that control the droplets, represent X values; typically they are 10 microns apart. Finally, the proportional deflection transitions themselves represent y values, and typically are in 10 micron increments.

To summarize, the location of droplets in the print bed are represented by addresses that look like Z.Y.X.y. An accuracy of 10 microns is available for both X and Y.y; for X it is every 10 microns, but for Y.y only 1 in 17 locations is addressable.

Abacus, then, represents a planar print region as a grid with an aspect ratio of 1:17, and each grid cell contains one marker to represent the proportional deflection of an individual droplet.

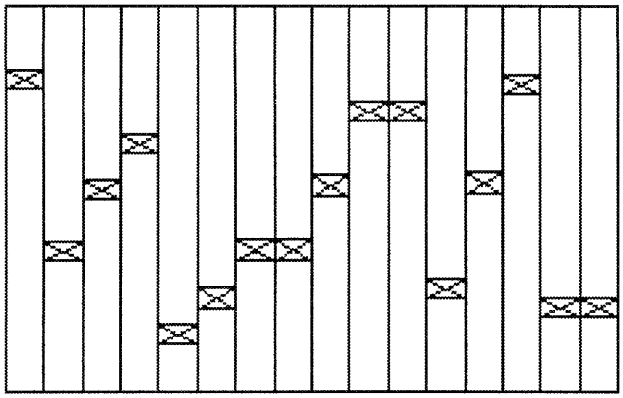


Figure 7.4 This is the Abacus unit cell, the individual cells represent the target area of the 3DP™ printing process, and the markers indicate the actual location of droplets.

The difficulty with this approach is that it is severely limited in its applicability. It can not be mapped to arbitrary surface regions. In fact, this method does not support arbitrary planar regions. In order to have a one to one mapping of voxels to print primitives there has to be a regular repetitious spacing of available print locations along the region to be textured. As the figure 7.6 indicates there is a limited number of available exact angles where for which there is such a rational, mxn, spacing. This introduces severe constraints on the part design process. While, it is possible to introduce leap rows, that the system could insert periodically, to increase the number of available angles, arbitrary precision is unattainable.

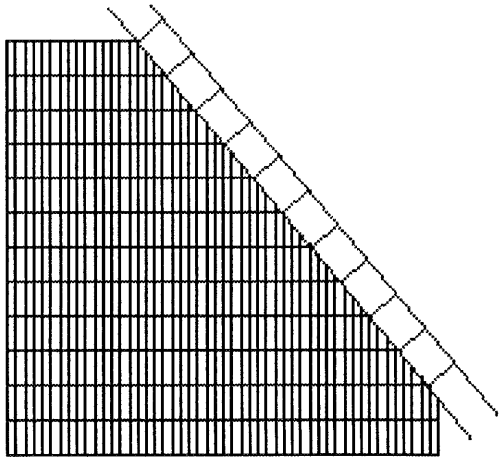


Figure 7.5 This shows the irregular cell spacing that occurs along a line drawn at an arbitrary location and orientation in the print bed.

In 3D space, the design process of a unit cell becomes untenable, for an arbitrary planar region:

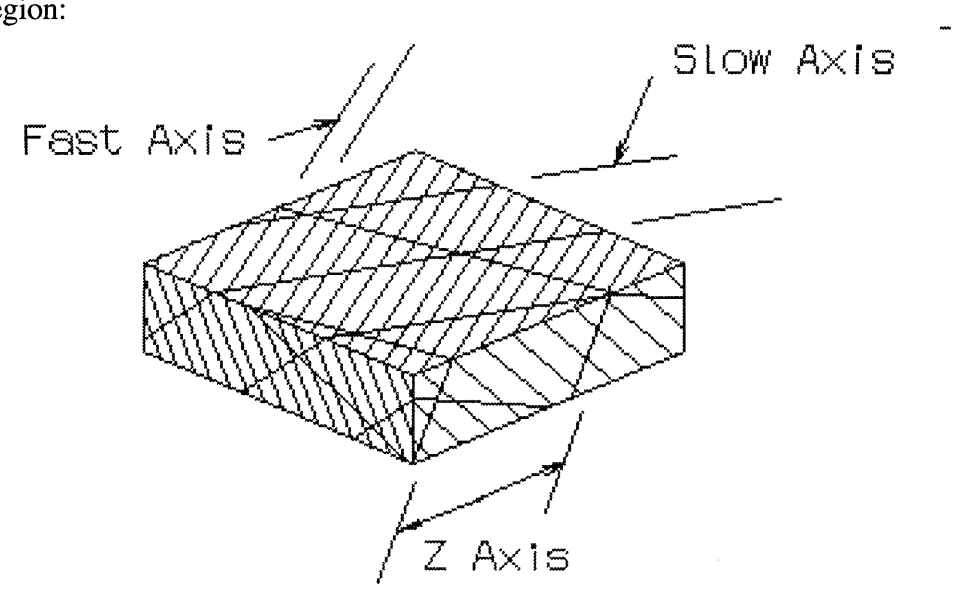


Figure 7.6 This is an arbitrary section of the print bed of the 3DP™ process. It demonstrates how complex such a region is to manage at a cellular level.

This unit cell concept is too specific to be of value to SFF community at large.

7.4 Swiss Solid Geometry

Swiss Solid Geometry, SSG, is a powerful and highly flexible representation for a unit cell, that evolved out of any earlier concept referred to as Swiss Cheese. SSG is scalable to different levels of resolution, at low resolution it can model functional texture for any SFF process, and at higher resolution it can model the features constructed with the printing primitives of the 3DP™ process.

7.4.1 Swiss Cheese

The initial concept of Swiss cheese is based on the idea that in the 3DP™ printing process, negative features can be arbitrarily small. It requires only a small disturbance of a few primitives to create a void.

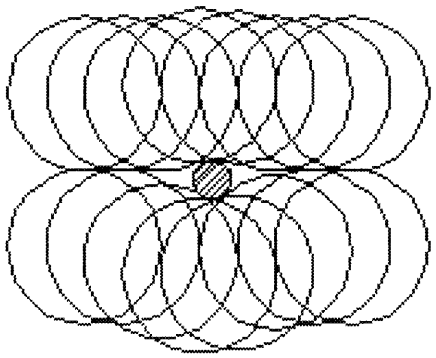


Figure 7.7 This shows how the small displacement of a few printing primitives, creates a void in the print bed.

This leads directly to the idea that the negative spaces should be explicitly modeled. It is very easy then to start with a filled region of primitive elements and push some of them around, in order to perforate the surface.

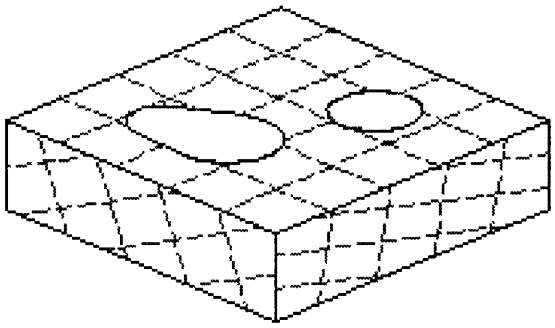


Figure 7.8 This graphic representation of the displacement of primitives in 3D is how the unit cell Swiss Cheese got it's name.

While interesting, this approach is limited by its inability to model positive features. Since parts and the tools that form parts, are negative images of each other, a designer may want to work back and forth between positive and negative modes. The important thing about this concept was that it was the first time that the idea of modeling the negative features is made explicit.

7.4.2 Positives and Negatives

Once both positive and negative regions are explicitly modeled it becomes much easier to manage the contiguity of the replicated cell. Cell editing schemes that treat the

negative region as the complement of the positive region, or vice versa; are much more difficult to verify in terms of manufacturing constraints.

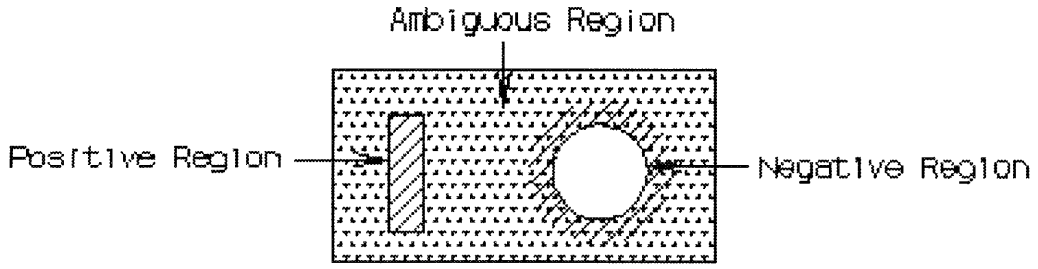


Figure 7.9 This is a 2D representation of the relationships between positive, negative, and ambiguous regions.

This unit cell consists of three basic entities: positive regions, negative regions, and ambiguous regions, that are undeclared. The applicability of the cell is manageable and visualizable, during the texture design process, by the allowance made between positive and negative features, ie. the extent of the ambiguous regions. The more ambiguity allowed the more flexible the unit cell, the less ambiguity allowed the less flexible the unit cell.

7.4.3 Primitives

7.4.3.1 Description

There are three simple primitives, and their offset regions. First there is the point, and the offset region around it is a sphere, whose radius is the offset. Next, there is a line, whose offset region is a cylinder. Then there is the convex quadrilateral planar region, and its offset region. A square is an example of the last, and when the offset is equal to 1/2 of the square's size; the offset region is a cube. When combined with boolean addition, this collection is very expressive, especially since some of the primitives are the boundaries of others. A flat ended cylinder is generated by a line segment, a capped cylinder is generated by a line segment that includes its endpoints. Unlike CSG, boolean subtraction is not included, and is not necssary since the negative spaces are modeled separately.

In 2D (the same is true for 3D), offset curves are conceptualized in two different, yet consistent ways. First, points on an offset curve are defined by the normal vector to the curve at some point, and the length of the normal vector is equal to the offset. Alternately,

the offset of a base curve is the envelope of all circles of fixed radius whose center is on the curve. For curves of infinite extent these definitions are fully equivalent. For curve segments the closure of the ends of the offset region is different. In SSG, the first concept is used, which allows the designer to choose the closure, by including specific elements for the boundary or not.

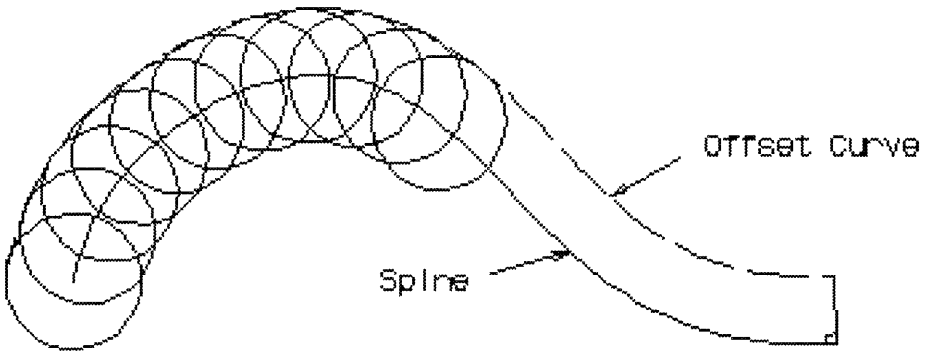


Figure 7.10 This shows the dual methods used to construct offset curves
(This also shows how printing primitives can be located by their centers.)

These primitive objects can be specified by points, one point, two points, or four points; which means that their locations and sizes are readily specified with geometric dimensions as discussed in Chapter 8.

While positive and negative tolerances are used for locations; the basic idea of SSG, that positive and negative regions are separated by ambiguity, means that only one sided tolerances are used for offset values. In order for a feature, positive or negative, to exist there must be a smallest offset, but to specify a maximum offset conflicts with the concept of the utility of the ambiguous region. The minimum offset for a positive feature is the size of the printing primitive, while the minimum offset for a negative feature is much smaller, as indicated by the original idea of swiss cheese. For the 3DP™ process, the minimum negative value is about three times larger than the powder size.

As the examples indicate, curves, and surface patches could be included; but some issues with the distances between such objects would need to be resolved

7.4.3.2 Motivation/Utility

A 3 axis milling machine positions a cutter and a work piece relative to each other, based on 3D information about the location of the tool tip. The position on the cutter that

represents a part surface is not relevant to the cutting of material, from the point of view of the machine tool. An NC toolpath, then, consists of a sequence of movements from one valid tool tip location to another valid tool tip location.

From the point of view of an NC CAM software, a valid tool tip location for a finishing operation is one for which the tool is just making contact with the part surface. The CAM software selects contact points, in order to trace the geometry of the part, and then calculates the tool tip location that corresponds to each contact point.

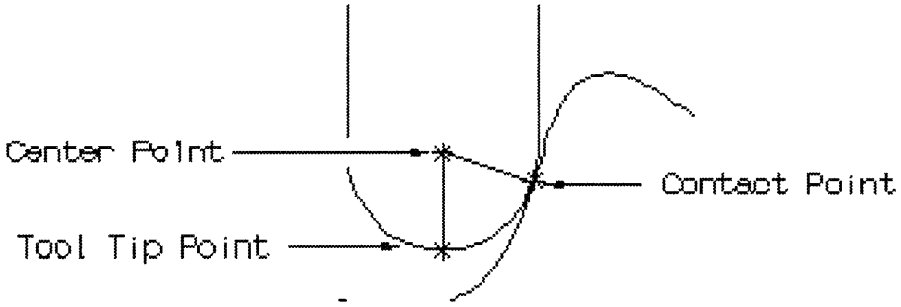


Figure 7.11 The analytic geometry involved in the calculation of a tool tip point from a contact point, in 3 axis machining using a ball end mill.

Specifically, for a ball end mill, it calculates the normal vector to the surface at the contact point, and then using the radius of the cutter finds the center point of the ball that forms the end of the cutter. The center point is then translated down in Z by the radius of the cutter. The final shape of the tool path is an unusual looking distortion of the part geometry.

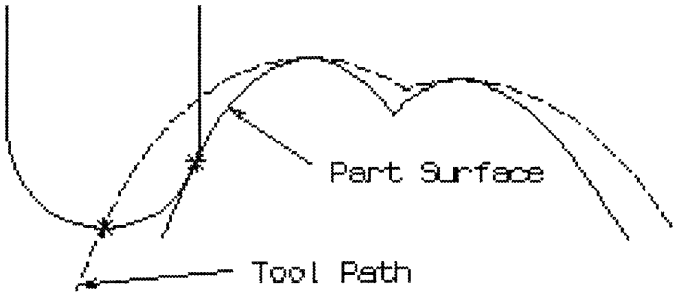


Figure 7.12 This shows the unusual shape of the tool path relative to the part surface from which it is generated.

I, like many NC programmers, have had difficulties verifying my tool paths.

Typically, we were taught to select points at random, or in regions of concern, and ditto a copy of the tool geometry to that location; then we intersect the tool geometry with the part geometry. If the result of the intersection is one point of contact, then it is a good point.

Seeking an alternative to this tedius process, especially for 3D freeform surfaces, I developed the following technique. I realized that the dual nature of surface offsets, as shown in figure 7.10 means that the center point of the ball end of the cutter is always located on the offset surface of the part surface, for which the offset matched the tool tip radius. In order to verify a tool path, I calculate the offset surface of my part surface as shown in figure 7.13,

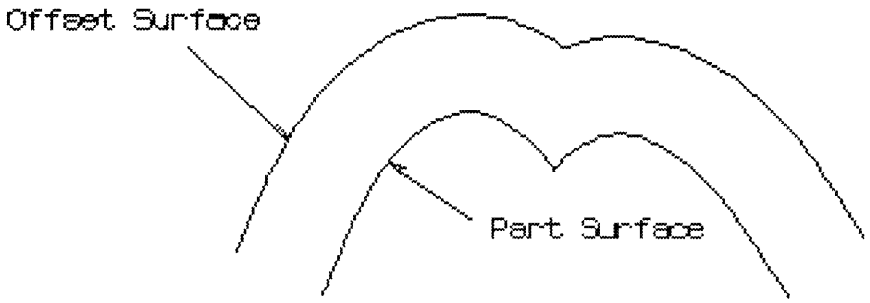


Figure 7.13 An offset surface calulated from a part surface,
which is a step in the process of determining a "tool tip" surface.

Finally, I translate the entire offset surface down in Z, by the radius of the cutter. This "tool tip" surface then has the exact shape, that the tool path has stepwise approximated. Any point on this surface is a valid tool tip location, and furthermore any curve on the surface represents a valid tool path.

The point is that a difficult problem in terms of the contact relationship between spheres and part surfaces is greatly reduced by the dual nature of offsets as presented in figure 7.10.

Amoung the many benefits of this simple relationship are:

The distance between two offset regions can calculated as the distance between the center, or spine, of the two offset regions minus the offset value of each region. This is much simpler because the entire geometry from the unit cell does not need to be processed by an FFD, only the central spine of the geometry is needed.

Interferance testing between objects is a simple extension of the above diatance calculation, a negative distance implies interference.

Inclusion testing, to see if a particular point in 3D space is included within such a primitive is also reduced to testing the distance from the point to the spine of the region to be tested for.

SSG is capable of fully modeling the functional elements of a unit cell, capturing the entire design intent, that is usually only implicit in the creation of positive and negative regions.

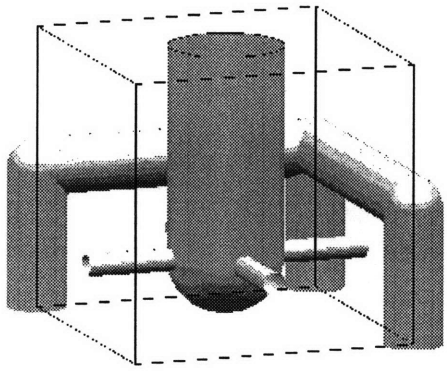


Figure 7.14 This is a 3D unit cell of Swiss Solid Geometry.

7.4.4 Disambiguation

While the ambiguity is useful as a modeling tool, it must eventually be resolved before the part reaches the end of the pipeline. Computationally, there are different methods by which the SSG representation can be utilized to calculate lower level information. The designer should select this method as part of the process of designing a unit cell of texture with SSG. While maximum offsets are not allowed, these modes offer additional control of the final result.

The inclusion test mentioned in the last section, can be utilized to treat a region as a gauge. A region used as a guage, is not used to create geometry, but it is interference checked with the other geometry, to see if it "fits" between the other regions.

The first mode occurs when the primary intention of the design, is to model positive features. In this case the calculations can proceed as follows:

First the 3D region on the surface is treated as empty, and the positive features are

built on this region. Then the negative features are used as gauges to check the manufacturability of the intended functional texture.

When the primary intention of the design is to model a pore space, then the computation would be: The region near the surface is closed in so as to be interior to the part, and the negative features are used to build geometry that perforates this region. Then, the positive features are used to verify the integrity of the resulting pore space.

There is a third mode that is particular to the 3DP™ process, that captures the high resolution of the 3DP™ process.

This technique is applicable to SFF as well as the 3DP™ process. For SFF an STL file can be written for the final textured surface, with the assurance that it is manufacturable regardless of method; providing that correct parameters are used to limit the feature sizes within the unit cell.

7.4.5 Applicability to the 3DP™ Process

In the 3DP™ process, as was mentioned in section 3.2.2 the print primitives are conceptualized as spheres. This makes the use of the offset space natural; in fact, figure 7.10 is a reasonable representation of the actual printing process.

There is a third mode that is particular to the 3DP™ process, that captures the high resolution of the 3DP™ process. This mode consists of examining each available print location within the ambiguous region. A dilation function can be used to determine whether or not a droplet should be placed at that location, based on the ratio of the distances of that print location to the nearest positive feature and to the nearest negative feature. The designer controls this process by selecting a desired ratio for the function. In this mode the positive and negative features swell to fill the entire region and do become mutual complements. To ensure manufacturability this would be done after an interferance check of the positive and negative regions.

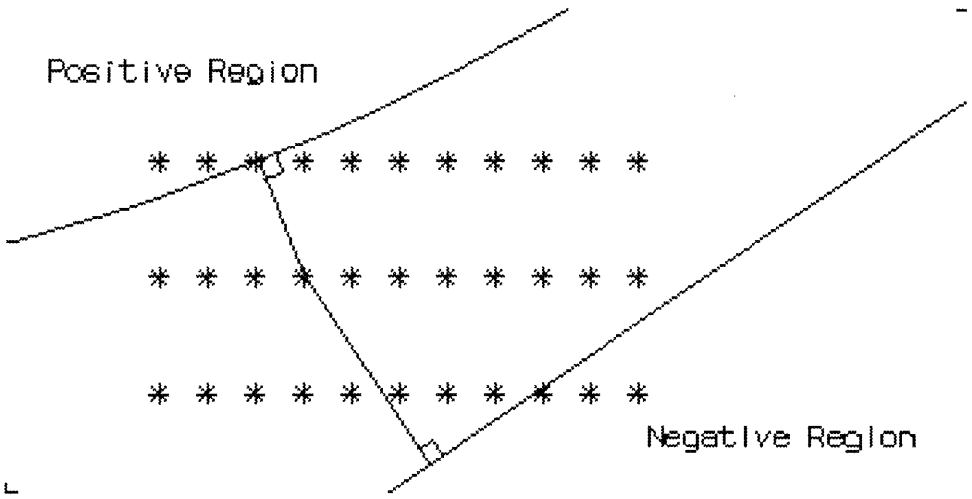


Figure 7.15 This shows how print decisions are made for each print location by the Dilation Principle.

7.4.6 Applicablity to VLSI MCAD

SSG is a tool that fullfills the separability requirement of VLSI MCAD, while being fully descriptive of both design and fabrication information. A functional surface texture, such as velcro can be described by a positive region resembling a hook, and a negative region can be included to ensure that the hook does not "close up" during fabrication. A product designer, though, does not have to be aware of the negative feature, nor even all of the positive features, in order to use velcro from a design library. That information, however, is carried along to the fabrication process, so that velcro will be accurately fabricated.

This capability comes from the behavior of the geometric primitives, that these primitives are directly specified by geometric dimensions; and from the explicit representation of positive and negative features. A complete description of a unit cell of functional surface texture consists of a list of dimensional relationships; and such a list is easy to separate into functional groups.

SSG is capable of representing many things. Physically isolated pore spaces can be built within such a unit cell, and capallaries with specific directions and topologies can be maintained, all while manufacturability is insured.

8. Dimensional Tolerances

8.1 Trivariate Free Form Deformation

Free Form Deformation is a technique of Computer Aided Geometric Design to extend the range of objects that can be described by Constructive Solid Geometry.

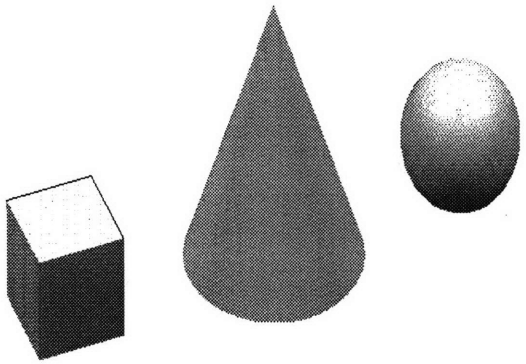


Figure 8.1 These are some of the primitive geometric objects used for Constructive Solid Geometry.

This technique utilizes bezier maps from a parameter space to the object modeling space. Typically, bezier maps are used to represent surfaces in a 3D CAD system. These surfaces are characterized by Control Points, and each point on the surface is a weighted average of the control points. The parameters u and v are the weights, and since the bezier map is continuous; the mapping of the unit square in parameter space, results in a continuous 3D surface.

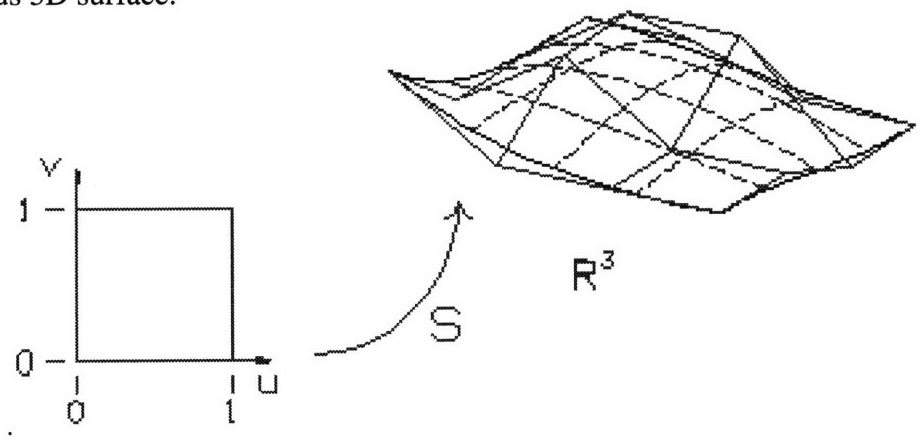


Figure 8.2 This shows how a Bezier map is used to represent a surface, from control points.

When the parameter space of a bezier map is Trivariate, 3D, then the map represents a transformation of 3D space. The control points of the map make these transformations easy to visualize. Furthermore, the convex hull property of the bezier map insures that the region contained within the control points in the parameter domain is mapped to the region contained within the control points in the object space.

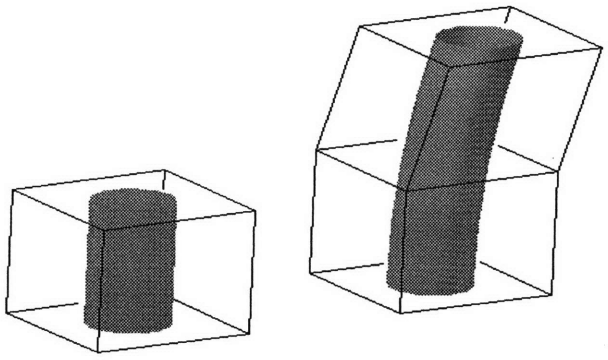


Figure 8.3 This shows how Free Form Deformation is used to shape simple objects into more comlex objects.

Obviously, geometric forms are distorted; under the mapping, lines are not straight and circles are not circular. This distortion complicates the prediction of distances in the object space from the parameter space. The best method to measure the objective distance between two points in parameter space, is to map those points into the object space; and then apply the euclidean metric to the transformed points.

8.2 Utilization for Texture Mapping

In this project, texture is created by replicating a unit cell of texture throughout the cells of a surface mesh. Each mesh cell is a separate Free Form Deformation of the unit cell.

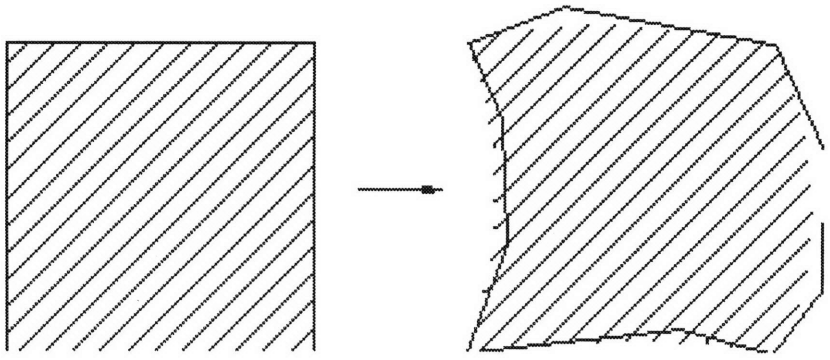


Figure 8.4 This demonstrates how a single unit cell is copied in order to fill a mesh.

This is achieved by creating an FFD, for each mesh cell, that has two control points in each dimension. A one to one relationship is established between the corners of the unit cell, and the corners of each mesh cell.

This particular utilization of FFD has an additional benefit. The degree of a bezier map is 1 less than the number of control points. In general the convex hull property is a bounding principal, and is not an exact statement.

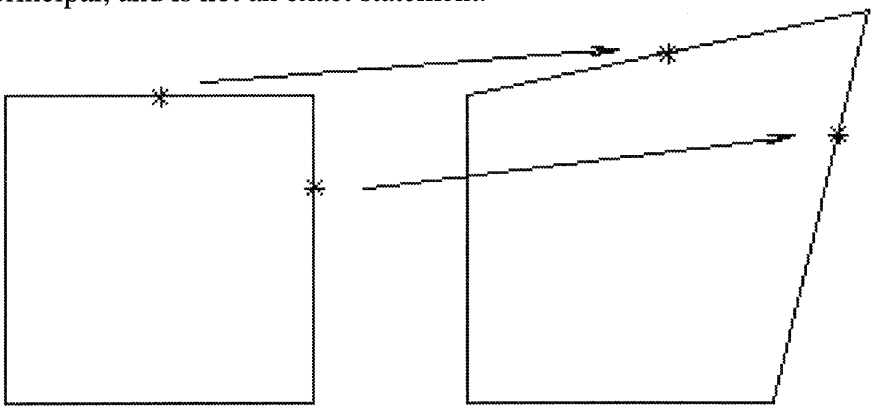


Figure 8.5 This demonstrates how the interior of a region is mapped by FFD, the edge of the pattern does not match the boundary of the control polygon.

However when the degree of the map is 1, the convex hull property is stronger. For this case the boundary of the unit cube (square) is mapped directly to the surface of the control polygon; and the interior of the unit cell is directly represented by the interior of the control polygon. In 2D, the control polygon is formed by connecting the control points with line segments.

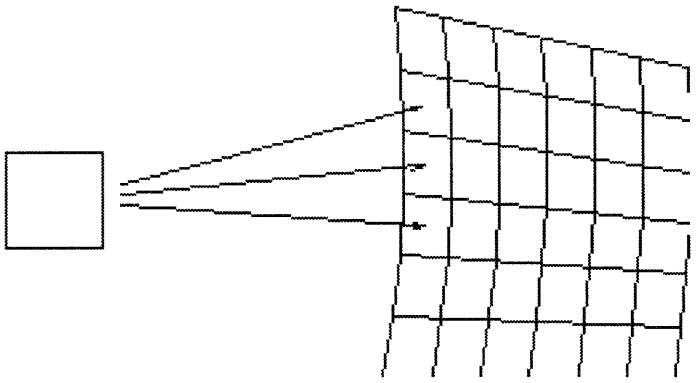


Figure 8.6 This shows that for an FFD of degree 1, the boundary points of the initial cell do get mapped to the boundary of the final cell

Generally, in 3D, the faces of the control polygon are not planar; but the boundary property remains true. This property is important, because it insures that when a unit cell of texture is replicated into the adjacent cells of a surface mesh (see Chapter 9. Meshing) the resulting texture is contiguous.

8.3 Separating the Dimensions from the Geometry

Dimensions are marked on the drawing of a part in order to capture design intent. These dimensions represent the important features of the part. Since some variation always occurs during fabrication, tolerances are established to set limits on the range of acceptable finished part dimensions. These tolerances are determined so as to insure the fabrication of functional parts.

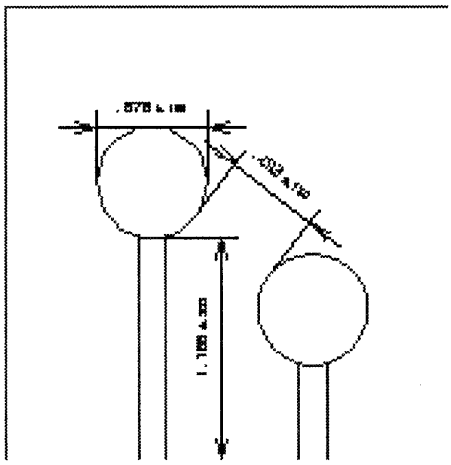


Figure 8.7 This is a representative technical drawing showing the usage of geometric dimensioning and tolerancing.

In process planning, these dimensional tolerances are used to determine the appropriate methods of part fabrication. The tolerances are compared to the constraints of different processes in order to choose among them.

The creation of an accurate, functional, surface texture requires these same considerations. Since the FFD distorts each copy of the unit cell differently, each mesh cell needs to be examined to see if it is within tolerance.

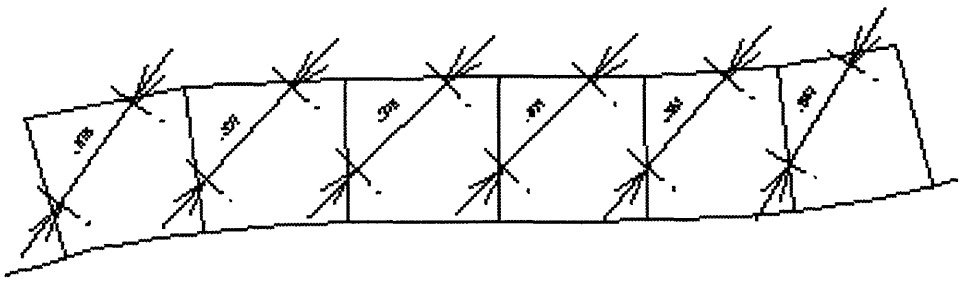


Figure 8.8 This shows how geometric dimensions vary when they have been copied by an FFD into a mesh.

In a machine shop, finished parts are inspected by measuring the size of and location of certain critical features. Good parts are those for which these measurements are within the tolerance range specified on the drawing. This inspection does not consider every nuance of the geometry, especially for parts that have some freeform surfaces; only the values of these measurements are important.

A similar inspection can be implemented to validate the quality of the mesh cells, if the geometric features in the unit cell have been dimensioned and toleranced. This inspection would be relatively quick because it does not require that all of the geometry be mapped from the unit cell to the mesh cell, just the dimensions need to be mapped.

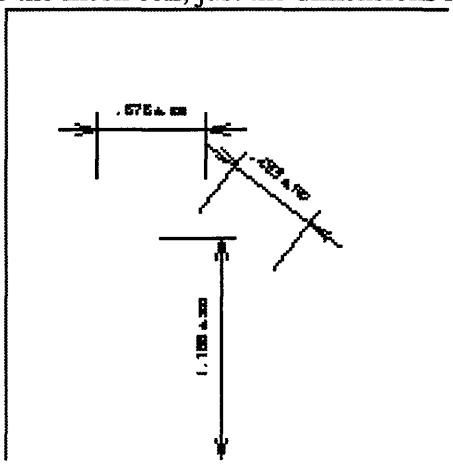


Figure 8.9 This is a conceptualization of the definitive information that is contained in a unit cell, ie the dimensions, by themselves.

8.4 Logical Consistancy of Constraints

If a dimensioned drawing is stretched in one direction then all of the dimensions parallel to that direction are stretched consistantly, by the same proportion. Dimensions that are perpendicular to the stretching are undisturbed, and so they have the same logical

consistancy that they had originally.

Dimensions that are at an angle to the stretch will have an amount of distortion that is between the proportion of the stretch and none. Logical consistancy is again preserved because the original dimension can be written as a linear combination of unit vectors perpendicular to and normal to the stretch; and the final dimension is the same linear combination of the unit vectors after the streching.

The FFD is an area preserving transformation. Logical conditions such as inclusion, exclusion, and lying on the boundary; are preserved under such a mapping. This logical relationship between dimensions, means that parallel dimensions are redundant in their ability to represent the stretching of the mesh cell. Ultimately, this means that only a few dimensions need to be specified to indicate the correct behavior of the mesh cell; and alternately that operations performed with the entire set of dimensions will not violate the few.

The decoupling of the dimensional tolerances of a geometry from the actual geometry, has several benefits. First, design rule checking can be performed with an element of abstraction that does not involve the complexities of a particular unit cell modeling paradigm. As long as the key dimensions can be specified, virtually any type of unit cell model can be utilized with some assurance of manufacturability. This element of abstraction facilitates meeting the requirements on MCAD to be VLSI-like. It also, greatly enhances the modularity of software implementations of functional surface texturing.

9. Mesh Regularity

As the previous chapters have explained, functional surface texture is created when a unit cell of texture is copied, by FFD into the cells of a mesh that lies on a surface. The accuracy of the finished texture is determined by the extent to which a mesh cell is not shaped exactly like the unit cell. To insure the quality of the texture, a mesh is required for which each cell is as much like the unit cell as possible.

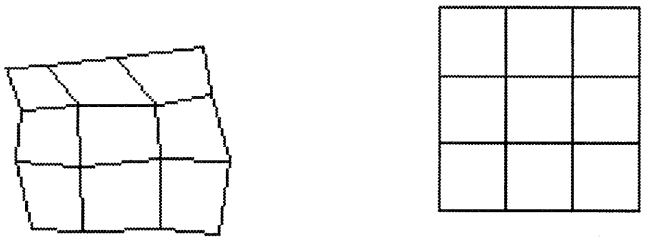


Figure 9.1 This compares an irregular mesh to a regular mesh.

Haeseong approached this problem by utilizing a physical analogy. He treated the mesh as though it were a box spring. Once he connected the mesh nodes with ideal springs, he was able to formulate a total elastic potential energy for the mesh. Then, his routine seeks a configuration of the mesh, such that this potential energy is minimized. This is an indirect approach to the ultimate problem of ensuring texture quality, in that it is after the mesh is regularised, that it is examined for accuracy.

A more direct approach is to use the functional conditions on each cell in the regularisation algorithm. Feed back is immediate, and is available to other algorithmic processes that could utilize this functional information to make other adaptations.

9.1 Terminology

There are alternate ways to visualize the mesh, and it is sometimes necesary to shift perspective. In 2D, a mesh can be viewed as an organized collection of nodes or as an ordered set of cells.

Figure 9.2 There are two views of the elements of a mesh, the nodes and the cells.

There is a consistancy to the different perspectives. On the one hand, each mesh cell is delineated by four corners, mesh nodes; and adjacent cells have some mesh nodes in common. On the other hand, successive pairs of mesh nodes define a mesh cell.

The best data representation is to store the 3D locations of the nodes in an array, that has an index for each direction that the mesh spans. In this project, the mesh considered is one layer of cells, so that there are two layers of nodes.

9.2 Classic Optimisation Algorithms

Generalized theories of optimisation are formulated in terms of an objective function for which an optimum value is sought. These algorithms are iterative, they start in some initial state, Xo, and then progress through a series of states, Xi, to reach a solution, Xf. Typically, X is a treated as a vector in an n dimensional vector space.

This progression of states is computed by analyzing the objective function to calculate a change to X, dX. If correct dX's are found at each step, then the sequence of states Xi converges to the solution. Divergency is characterized by one of two conditions: the state "blows up", the magnitude of Xi, |Xi| increases toward infinity; or the states oscillate, a sequence of states, Xi..Xj, repeats.

The analysis is based on the methods of multivariate calculus, the minimum is at a location where the derivative of the objective function is zero; and entities such as Jacobian and Hessian matrices are calculated in order to determine the direction and magnitude of dXi in n dimensional space.

9.3 Displacement Algorithms

Manipulations of an n-vector are not readily visualizable for n greater than three. Since, the classical algorithms rely heavily on the theoretical basis of calculus to acheive their results; there is little room for intuition.

In some formulations, such as this project's surface mesh, the dX changes between states have a natural geometric interpretation. The n dimensional vector, dX, can be partitioned into 3D displacements for each mesh node; dXi is readily visualized as the change in locations of the mesh nodes between steps Xi, and step Xi+1.

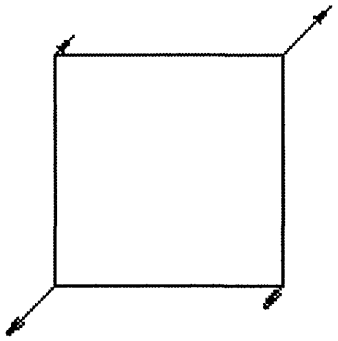


Figure 9.3 Each vector is a mesh node displacement,
and each is a partition from the generalized n-vector, X.

This conceptualization, also, asserts itself in the implementation of a mesh regularization algorithm. A first partition of X, occurs when the mesh nodes are stored as 3D values in the array described above. A further partitioning is implicit in the topology of the mesh, the displacement of an individual node is based on its relationship with the nodes to which it is connected. The total displacement of a mesh node is an average of individual displacements that are calculated by looping through the mesh and the pairwise comparison of the current node with its surrounding nodes.

An alternative to the classical approach is available in this situation. Instead of formulating the algorithm in terms of an objective function, and then deriving dXi; it is possible to formulate the algorithm directly from the easily visualized displacements. As long as the displacements meet the conditions of direction and magnitude, convergence will be assured. In the classical formulation convergence is observed as the diminishing of the magnitude of Xi, in the displacement formulation the convergence is visible as the diminishing motion of the nodes. The objective function does not need to be explicitly

stated. The nodes will move to stationary locations even if individual residual displacements aren't all zero, once the accumulated displacement for each node is zero. This is equivalent to the classical situation when the minimum of the objective function is not at zero.

The primary condition on displacements for convergence are that the nodes move with progressively smaller steps. The condition on the displacements to avoid "blowing up" is that the steps are never so large as to fold the mesh.

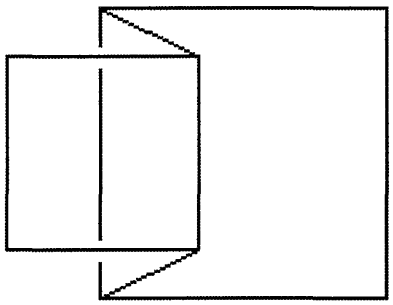


Figure 9.4 This shows how a mesh can be folded
if the nodes of adjacent cells are pushed past each other.

Along with easy visualisation comes the facility to use physical analogies. Any devised displacement can be modeled as a force, and the square of any displacement can be modeled as potential energy; by associating a spring constant to the displacement. If desired, an object function can always be constructed. It can be useful to work between the classical point of view and the displacement point of view.

In order for the algorithm to not oscillate the system needs to be stiff, slow, or dampened.
In terms of displacements these are all equivalent to limiting or scaling the size of the displacement.

This is an alternative way to view Haeseong's energy minimization routine.

9.3.1 Example 1 - Relaxation

Relaxation is a technique that was originally devised to calculate stresses in structures. Given an initial framework, the distortion of the framework when a load is

applied is calculated by progressively reducing the residual forces in the joints of the structure.

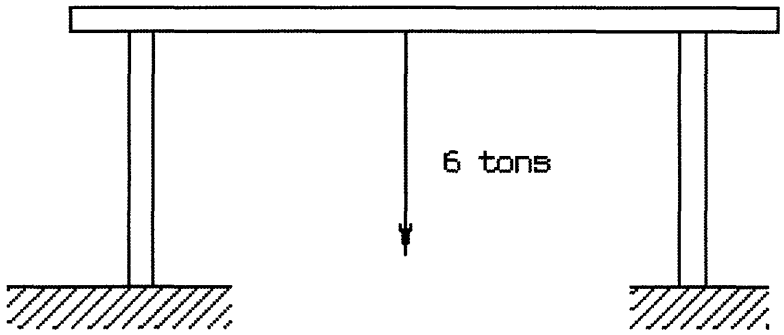


Figure 9.5 This is a representative structural problem.

At each stage of the calculation, the largest residual stress is reduced by calculating appropriate pin joint displacements. The pin joint locations converge to final locations, even if the residual stresses can not be totally eliminated.

The final shape of the structure can be solved for, even though an objective function has not gone to zero.

9.3.2 Example 2 - Iterated Function Systems of Contractive Linear Transformations

IFS is a technique that has been developed to compress computer graphic images. It uses the property that the orbit of a point under the influence of a collection of linear transformations can condense to a compact set, under the condition that all of the transformations in the collection be contractive. [BAR]

Contractive refers to the property that the net scaling of the transformation is less than 1. The images of X_0, $A^N.X_0$ as N goes to infinity is a finite point, a limit point.

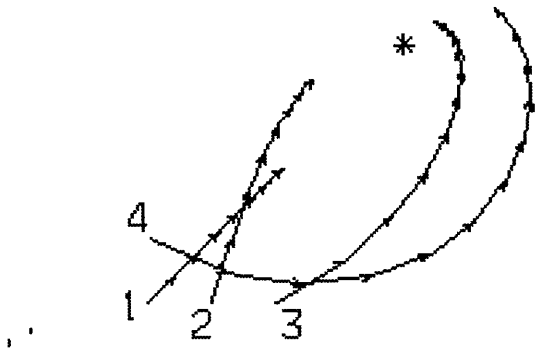


Figure 9.6 These are examples of contractive linear transformations, and the orbit of an arbitrary point under their influence, as they move to a common limit point.

1. An isotropic scaling.
2. A nonisotropic scaling, the x scale does not equal the y scale factor.
3. A nonotropic scaling combined with a rotation.
4. An isotropic scaling combined with a rotation.

Another example is a linear affinity, which is a transformation that pulls points to a line, along the normal vector from the point to the line.

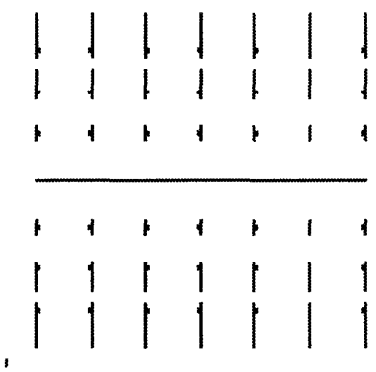


Figure 9.7 This is a linear affinity with the orbits of several points are marked.

The interesting thing is that a similar property exists for a collection of such transformations. The orbit of a point under the influence of a collection of contractive affine transformations converges to a compact point set; even though the transformations are applied in random order. The limit set of points is refered to as a chaotic attractor.

Figure 9.8 This is the compact set of points that an point in the plane will move onto under th influence of an IFS. It is a Chaotic Attractor.

9.4 This Algorithm

An algorithm has been developed for this project, that utilizes the multiple copies of the dimensional tolerances, that are shown in figure 8.8. The dimensions contained in each mesh cell are used to calculate displacements for the mesh nodes that are the corners of each cell.

Once the value of the dimension has been determined by calculating the distance between the limit points that mark the ends of the dimension, that value can be compared to the nominal value of the dimension as declared in the unit cell. The gross displacement of the limit points that is needed for the measured value to match the nominal value is calculated next. For each of the mesh cell's corner nodes, this gross displacement is scaled according to the parallel distance that the node is from the central point between the two limit points.

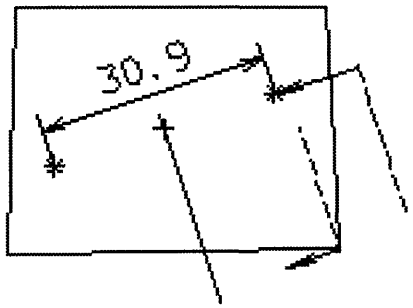


Figure 9.9 The displacement calculation of a node based on the correction needed for dimensional accuracy.

This calculation is equivalent to the linear affinity show in figure 9.6.

The tolerances are used to dampen the process. When a dimension is within tolerance, it is turned off, so that it does not disturb any nodes. However, if the nodes are disturbed by the dimensions in a neighboring cell, then it can become active again. The width of the tolerance range of the dimension determines the amount of disturbance that the current mesh cell can absorb.

This absorption process dampens the algorithm, and can provide a shortcut stopping mechanism, in those cases where there are no residual stresses.

While not implemented this way, it should be possible to make this algorithm behave entirely like relaxation, by starting with arbitrarily large tolerances, so that only the worst dimensions are active, and then progressively tightening the tolerances down to the designer specified values.

10. The Implementation Specifics of Vari_4

The texture modeling techniques discussed in this work are implemented in a software called Vari_4. Vari_4 is the name of the main program in C, that the functional modules plug into, as in figure 10.2; and it is the name of the executable software. This is a modal software, in the sense that one 3D window is used to manipulate either the part geometry or the cell geometry. These modes are conceptually related, but in fact represent different spaces. The cell mode is like a geometric detail that is to be dittoed into a master workspace, the part mode.

10.1 Modularity

There are many well known reasons to write computer code in a modular style. A modular style was utilized for this project in order to make this program extensible. It should be possible to add functionality to the base software, continuously; as new concepts and new technologies are developed by the students and staff of the 3DP™ Consortium.

The modular style of the computer code was useful during the system development, also. Modules were developed and tested in isolation from the main program, which simplified the debugging process.

The program is organized so that each module represents a different operating functionality of the overall design process. In this structure, each module is responsible for a particular data structure.

Module	Data	Initialization	Geometric Display	Parameter Control
surf_2	CP		draw_SURFACE	
axis_2	AXIS	build_AXIS	-callobj-	
mesh_4	MESH		draw_MESH	install_MESH
				pop_MESH
unit_2	UNIT	build_UNIT	-callobj-	install_UNIT
				pop_UNIT
dimen_4	LIMIT	initialize_DIMENS	draw_DIMENS	install_DIMENS
			map_DIMENS	pop_DIMENS
patt_2	PATTERN	initialize_PATTERN	draw_PATTERN	install_PATTERN
			map_PATTERN	pop_PATTERN
slice_4		initialize_SLICE	draw_SLICE	install_SLICE
			draw_HATCH	pop_SLICE

Table 10.1 These are the modules that plug into the main program, Vari_4.

A limited form of version control is used in the naming of the modules, mesh_4 is the fourth version of the MESH handling routines.

A naming convention is used to name particular functions in each module. This introduces some commonality to each of the modules. Some of the geometric objects are static and some are dynamic. The static objects are registered as Gl objects with build_ routines and then those objects are displayed by the Gl routine, callobj. The dynamic objects are displayed using a draw_ or a map_ routine. A draw_ routine is used to display an object in it's native mode, part or cell. A map_ routine is used to display an object replicated throughout the cells of the mesh, in space mode. The dynamic objects are initialized by initialize_ routines. The dynamic geometric objects allow the user to modify their key parameters through the use of popup windows. A popup window is established with an install_ routine, and it is displayed, or put away, with a pop_ routine.

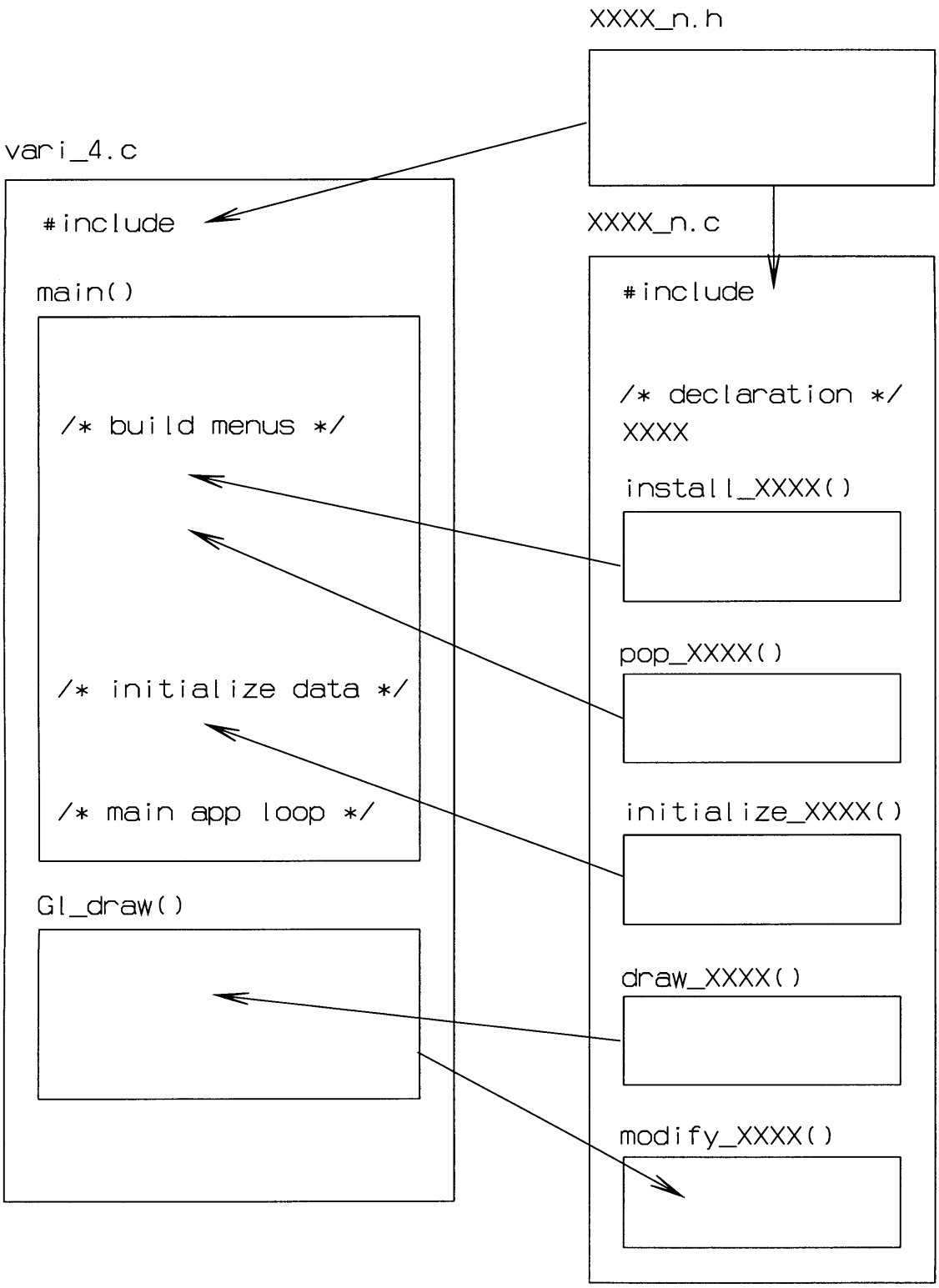
10.2 Module Integration

Vari_4 and the functional modules are structured so that the functions can be "plugged in" to the main program. See Figure 10.2 Program Structure, on the next page.

An arbitrary module, XXXX_n.c, is shown in figure 2. The header file XXXX_n.h, is structured so that the data structure XXXX is declared in XXXX_n.c, and defined as EXTERN to the main program. In the most complete case, that XXXX is a displayable geometric object with modifiable parameters, XXXX_n.c is responsible for the five particular subroutines shown. The boxes show where the functions are coded, and the arrows show from where the functions are called.

The main program manages the interactive behavior of the application. It provides a menuing system for the modules to hang their controls, it keeps track of the visibility of the different geometric objects, and it converts mouse input in the graphic region into graphic motions. While the main app loop handles the menuing, Gl_draw() handles the graphic display.

Since, every geometric object has it's own unique visual quality, each module provides an appropriate function to display the object; that it "registers" with Gl_draw(). Then, Gl_draw is made public by Vari_4, so that as the parameters of the geometric object are modified, the graphic display can be updated.


XXXX_n.h
vari_4.c
#include
main()
/* build menus */
/* initialize data */
/* main app loop */
GL_draw()
XXXX_n.c
#include
/* declaration */
XXXX
install_XXXX()
pop_XXXX()
initialize_XXXX()
draw_XXXX()
modify_XXXX()

Not shown in figure 10.2, is the internal linkages within vari_4.c and XXXX_n.c. In Vari_4, Gl_draw() is called by the graphic area's input callback function; so that the display updates in response to mouse input. In XXXX_n.c, modify_XXXX() is usually called by the popup window that is defined in install_XXXX().

10.3 Module Descriptions

Points are represented by the data type, pt_type_3, which is a structure of three floating point values, X, Y, Z; declared in points.h.

Misc contains four functions that facilitate the usage of TextField with numeric values:

get_int and get_dbl take handles to TextField widgets and return numeric values.

is_int and is_dbl takes handles to TextField widgets and tests whether or not the TextField represents a valid numeric value.

10.3.1 Mesh

The data structure of the MESH is an array[i][j][k] of points. The indices, i and j, span the region to be textured, while the index, k, is normal to the region. The index, k, is allowed to have only two values; k = 0 is the set of mesh nodes that are on the surface, and k = 1 is the matching set of nodes that are offset from the surface.

draw_MESH displays the mesh by drawing line segments between adjacent node points of the mesh.

The mesh popup controls the size, nxm of the mesh, by calling do_MESH_size_OK, which calls initialize_mesh and gl_draw. The popup can also execute two different mesh regularization routines; energy_MESH which is an implementation of Haeseong Jee's energy minimization routine, and constrain_MESH which is the implementation of the algorithm described in Chapter 9. The regularization routines are readily adaptable to different surface types by replacing the single call in each of surface_projection.

Mesh also supplies the functions:

transform_t, which is the implementation of the Trivariate Free Form Deformation discussed in section 8.1. It used by the routines map_DIMENS, and map_PATTERN.

10.3.2 Pattern

PATTERN is the Checkerboard model of a unit cell of texture, described in section 7.2. The data structure is an array[i][j][k] of 0's and 1's, stored as integers. This scheme facilitates conditional operations through the usage of the existential state value, 1.

The pattern popup controls a layer of the PATTERN with a grid of voxels that the user can interactively select and deselect. Up and down arrows navigate the layers that comprise the third dimension of the PATTERN.

draw_PATTERN renders the PATTERN in the cell mode, which is 3D, by displaying polygons for each of the six faces of an instantiated voxel.

map_PATTERN renders the PATTERN throughout the cells of the MESH.

For both draw_PATTERN and map_PATTERN, faces are suppressed between adjacent active voxels.

Pattern also supplies the routine:

write_PATTERN which writes out an STL file of the PATTERN, where each of the six faces of an active voxel, are split into two triangles.

Internally, pattern uses the routine initialize_CORNER to calculate the 3D floating point values of the corners of the voxels, based on the physical dimensions of UNIT, and the nxmxk subdivisions into which the PATTERN breaks the unit cell.

10.3.3 Surf

The public data of the surface module represents the control points of a bezier surface.

draw_SURFACE uses the Gl function nurbssurface() to display the surface. The knot sequences are simplified to nx0's followed by nx1's to get the intended behavior.

Surf also supplies the functions:

surface_point takes in u and v parameters and returns a point on the surface.

surface_norm is an extension of surface_point that returns, in addition, the normal vector to the surface at the calculated point. surface_point and surface_norm are used by initialize_MESH.

surface_projection takes an arbitrary point in space and calculates the surface point that is the projection of that point. It also accepts and returns the u and v values of the calculated point. The accepted u and v values are used to seed the iterative algorithm.

surface_projection is used by the mesh regularization routines energy_MESH and constrain_MESH.

Internally, surf uses the functions bezier and beztan, which are implementations of Aitkin interpolation to different depths.

10.3.4 Dimens

The public data of the dimens module are two limit points that represent the caliper points of a dimension; and two tolerance values, the minimum and maximum allowable distance between the limit points. The nominal length, the distance between the two limit points as they are declared within the unit cell, is also stored, for computational convenience.

The dimens popup allows the limit points to be edited, calculates the nominal length, and then allows the tolerances to be edited.

draw_DIMENS displays the DIMENS in the unit cell as line segments drawn between the limit points.

map_DIMENS displays the DIMENS as line segments throughout the MESH. It also color codes the line segments: yellow for line segments whose length is within the tolerance, and red for those that are not in tolerance.

10.3.5 Unit

The public data of the unit module are the physical dimensions of the unit cell. The usage of the identifiers X, Y, and Z, is redundant to pt_type_3, but not the same data type.

The unit popup allows the dimensions of the unit cell to be edited.

10.4 Porting Path

This software is implemented on a workstation because of the demands of 3D graphics and because of the large file sizes, that can be encountered in SFF and in the 3DP™ process. Currently, Vari_4 runs on an IBM RS6000 with AIX 3.2.5, and an SGI INDY with IRIX 5.2.2. This is possible because these two environments support three things in common: the C language, Motif 1.2, and IRIS Gl.

In this regard, this application is directly related to the work of John Lee, who used C and Motif; and to the work of Haeseong Jee, who used C and IRIS Gl.

Porting this software to other platforms will require some upgrading of the code. The first step should be to upgrade IRIS Gl to OpenGl. OpenGl is supported on a large number of different workstations, as is Motif; so that Vari_4 could be run on many different workstations. Since Motif was used for the user interface and mouse events, only a small subset of the IRIS Gl are involved. A porting tool such as toogl, may be able to port this code on the first pass.

The next step along the path is not as clearly defined, and will be more involved. PC's running some version of Windows™ continue to gain in power and popularity; however replacing the Motif commands with Windows™ commands will be more difficult. Fortunately, OpenGl is supported by both Windows 95™ and Windows NT™.

11. Future

11.1 Alternative Surface Texturing Techniques

In Chapter 6, a particular approach to texture, based on certain concepts of functionality is chosen from the vast range of all descriptions of texture. Here are some other approaches to surface texturing, that could, under experimentation, demonstrate useful functionality.

11.1.1 Dithering

Points on the surface could be chosen purely at random, and then they would be perturbed normal to the surface by some randomly chosen value. This could be implemented as a post process applied to an existing file, for the 3DP™ process. The texture would be characterized by two parameters, the extent to which the algorithm is allowed to run, and the average perturbation.

11.1.2 Filters

The dithering approach could be modified with the use of filters. Digital images are processed by filters that examine a group of cells in order to modify an individual cell, typically one in the middle of the group. When the dithering process selects point location to modify, the nearby print locations could be "filtered" in order to determine a perturbation. Partially randomized patterns would emerge at larger scales, similar to swirls on a stucco wall; based on the "shape" of the filter.

11.1.3 Repeated Subdivision

One variation of meshing covers an irregular region with small triangles. At each stage of the process, it determines the size of each existing triangle and subdivides the triangle if it is too large. A simple pattern could be created by printing only those small triangles that were created by an odd number of subdivisions. More sophisticated patterns could be created by using rules that establish a state for each triangle at each stage, based on other properties of the subdivision process.

11.1.4 Cellular Automatons

In one sense, the approach to texture in this work has been a global process. The compensation for the curvature of a region to be textured is calculated before texture mapping occurs. A local process would walk along a region; and then based on the local curvature, and the history of what it has recently printed; would decide what to print next.

11.2 Alternative Cell Editor

While it is more natural to design unit cells that are four sided, the STL file format is based on triangles. A unit cell editor that produces useful three sided texture cells, would be able to capitalize on the existing STL format. In fact, the STL format allocates storage for two arbitrary parameters for each triangle that are seldom used. These parameters could be used as pointers into a table of a variety of three sided texture cells, that would be utilized during the printing process.

11.3 Incorporate Other Functionalities

Ideally, all of the new functionalities enabled by SFF and the 3DP™ process should be designed into a part with one software package. I would like to see Vari_4 be the ancestor of that software. The functionalities that will be the easiest to incorporate are those that are most similar to surface texturing.

11.3.1 Conformal Cooling Channels

In one sense, conformal cooling channels are subsurface textures. Small capillaries that are finely divided from larger capillaries, repeated just below a surface so as to create a particular amount of thermal transfer. The work of Xiarong Xu could provide design rules that could determine the appropriate design parameters for such a "texture".

11.3.2 Functionally Gradient Materials

Todd Jackson is working on new CAD representations for the structure of a solid object so that the microstructure of a part can be assigned throughout the interior of the part. The goal is to facilitate the creation of parts that are composed of Functionally Gradient Materials.

One of the approaches that Todd is considering is a volumetric meshing of the part, that includes the assignment of properties to individual volume elements. This approach to

functionally gradient materials could fit well with this projects approach to surface texturing.

11.4 Mesh Issues

11.4.1 Boundary

The mesh could follow the boundary better. The array that is being used to store the mesh elements, should be replaced with a data structure, that has two bidirectional linkages. The one linkage would be used to represent the adjacency in the u direction and one linkage would be used to represent adjacency in the v direction. This data structure would allow new cells to be added or deleted, around the edge, as the overall mesh contracts or expands, respectively.

The mesh would be able to follow more complex, even nonconvex, boundaries.

An interesting approach to mesh regularisation is suggested by this idea. The mesh could start with just one cell, and then in stages, cells could be added around the edge of the existing mesh. At each stage, the mesh is regularized. Since the first cell can be created in a regular fashion, any disturbance to regularity is encountered by the expanding edge of the mesh. If at some stage the mesh can not be satisfactorly regularized, then bad surface regions are immediately identified.

11.4.2 Dislocations

In nature, crystals organize into nearly perfect structures by introducing point dislocations in the structure. It would be interesting to rewrite Haeseong Jee's energy algorithm to pop out or pull in additional mesh nodes, if the local spring energy were to exceed some threshold value. Schemes would have to be found to collapse the data structure around such dislocations.

Index/Glossary

3DP™ Process - The Three Dimensional Printing Process - This is a registered trademark of MIT.

API - Applications Programmable Interface

APT - Automatic Programmable Tool

CAD - Computer Aided Design

CADCAM - a generic term used to discuss information issues, in a broader context than either CAD or CAM inplicate by themselves.

CAM - Computer Aided Manufacturing

CSG - Constructive Solid Geometry

ECAD - CAD for the design of Electronic devices

FFD - Free Form Deformation

Freeform is the usage by the SFF community
Free Form is the usage by the CAD community

FGM - Functionally Gradient Materials

IFS - Iterated Function System

LOM - Laminated Object Manufacturing

MCAD - CAD for the design of Mechanical devices

MIT - Massachusetts Institute of Technology

NC - Numerical Control

NSF - National Science Foundation

Part - Any single object. refers alternately to the physical and the CAD model ...

Post Process - Those processes that are required to complete the functionality of a part after a primary shaping operation.

Printing Primitive - The smallest feature printable by the 3DP™ process. Conceptually a droplet, typically represented as a circle or a sphere.

SLA - Stereo Lithography

SLS - Selective Laser Sintering

SFF - Solid Freeform Fabrication

SSG - Swiss Solid Geometry

Tooling - a collection of parts used to manufacture other parts A high volume process or to shape materials that are not readily cut.

Voxel - A Volume Pixel, which is an abstract concept for the smallest feature buildable, by an SFF process, or by the 3DP™ process.

VLSI - Very Large Scale Integration

VLSI MCAD - Mechanical CAD that behaves like VLSI

References

[ALL 54] Relaxation Methods, D.N. Allen
McGraw-Hill, 1954 (QA225.A425)

[ANG 87] High Resolution Computer Graphics Using Fortran 77, Ian Angell and Gareth Griffith, 1987 McMillan

[BAR 88] Fractals Everywhere, Michael Barnsley
Academic Press, 1988

[BEI 76] Applied Geometric Programming, Charles Beightler and Donald Phillips
1976 Wiley (T57.825.B44)

[COR 95] "Three Dimensional Printing of Ceramic Molds with Accurate Surface Macro- Textures for Investment Casting of Orthopaedic Implants", Alain Curodeau,
PhD Thesis in Mechanical Engineering, MIT, Sept. 1995

[CHU 74] Elementary Probability Theory with Stochastic Processes, Kai Lai Chung
Springer-Verlag, 1974

[HEN 90] Discrete Relaxation Techniques, Thomas Henderson
Oxford University Press, 1990

[HAR 88] Computer Graphics, A Programming Approach, Steven Harrington
1988 McGraw Hill

[JAC 92] Rapid Prototyping & Manufacturing, Paul F. Jacobs
Society of Manufacturing Engineers, 1992 (TS155.6.J33)

[JAH 93] Digital Image Processing, Bernd Jahne
1993 Springer Verlag (TA1632.J34)

[JEE 96] "Computer-Aided Design of Surface Macro-Textures for Three Dimensional Printing", Haeseong Jee
PhD Thesis in Mechanical Engineering, MIT, April 1996

[KUM 95] "An Assessment of Data Formats for Layered Manufacturing", Vinod Kumar, Debasish Dutta
to be published in Advances in Engineering Software, Elsevier Apllied Sciences

[LEE 96] "Computer-Aided Process Planning for Surface Quality in Point-Wise Constructive Manufacturing by Three Dimensional Printing", Sang-Joon Lee
PhD Thesis in Mechanical Engineering, MIT, Sept. 1996

[LUE 84] Linear and Nonlinear Programming, David Luenberger
1984 Addison-Wesley (T57.7.L948)

[NSF 96] "NSF Workshop on Design Methodologies for Solid Freeform Fabrication June 5-6, 1995"
Engineering Design Research Center Carnegie-Mellon University, 1996

[PRE 89] Numerical Recipes in Pascal, WH Press
1989 (Q76.73.P2.N87)

[RAV 90] A Taxonomy for Texture Description & Identification, A Ravishankar Rao
1990 Springer Verlag

[SAC 94] "Design Automation for Solid Freeform Fabrication", Emanual Sachs, Michael Cima, Duane Boning, Subra Suresh, David Gossard
Proposal to the NSF, June 1994

[SAH nn] "Heterogeneus Process Simulation Tool Integration", Zakir H. Sahul, Kenneth C. Wang, Ze-Kai Hsiau, Eugene W. McKenna, Robert W. Dutton
to be published, Stanford University

[SAI 94] "Meshing of Engineering Domains by Meitotic Cell Division", Kazuhiro Saitou, Mark Jakiela
Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, The MIT Press, July 1994

[SHI 92] "Computational Methods for Physically-based FE Mesh Generation", Kenji Shimada, David Gossard
Proceedings of the IFIP TC5/WG5.3 8th International Conference on PROLAMAT, June 1992

[TAH 71] Operations Research, Hamdy Taha
MacMillan 1971 (T57.6.T128)

[ULI 87] Digital Halftoning, Robert Ulichney
MIT Press, 1987 (T385.U45)

[WHI 87] Relaxation Techniques for the Simulation of VLSI Circuits, Jacob White, Alberto Sangiovanni-Vincentelli
Kluwer Academic Publishers, 1987 (TK7874.W48)