Medial Surface Transformations for Rapid Approximation of Casting Solidification

by

Scott A. Houser

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

APPROVED:

[illegible]

Dr. Jan Helge Bøhn, Chairman

[illegible]

Dr. Michael P. Deisenroth

[illegible]

Dr. Arvid Myklebust

July 1996

Blacksburg, Virginia

Medial Surface Transformations for Rapid Approximation of Casting Solidification

by

Scott A. Houser

Jan Helge Bøhn, Chairman

Department of Mechanical Engineering

ABSTRACT

This thesis demonstrates the feasibility of using a medial surface transformation as a tool to rapidly approximate the solidification patterns of convex faceted solid models of castings. The medial surface transformation is used to automate the greatest included sphere approach to solidification pattern approximation. The experimental software of this thesis extracts the medial surface transformation from a convex faceted model by computing the model's Voronoi diagram and uses it to identify casting hot spots and cooling patterns. Comparison with a finite difference method (FDM) solution showed that the locations and shapes of hot spots predicted by FDM converge to the shapes and locations predicted by the experimental software.

Acknowledgments

I would like to recognize and thank the following people for their contributions to this thesis:

- Dr. Jan Helge Bøhn, my thesis advisor, for his technical expertise and direction.
- My committee members, Dr. Arvid Myklebust and Dr. Michael Deisenroth, for taking time from their busy schedules in order to facilitate my efforts during both the academic year and summer sessions.
- Shawn Fitzgerald, for working with me on the early drafts of my thesis and for providing support and encouragement.
- Emmanuel Sabourin, for providing me with some technical advice on marching algorithms.
- The Virginia Tech CAD Laboratory and Darrell Early for providing the hardware and technical support needed to develop the software used in this thesis.

I would also like to acknowledge the Mechanical Engineering Department at the Virginia Polytechnic Institute and State University for providing me with teaching assistantships, and the Naval Surface Warfare Center, Dahlgren Division for providing research assistantship funding under contract N60921-89-D-A239, Order 0045.

Finally, I would like to thank my father for getting me interested in mathematics and science at an early age, my mother for teaching me enthusiasm and positive thinking, and my brother for teaching me that new challenges should be faced with relentless, maximal effort. I would like to thank them all for providing me the love and support that has been instrumental in granting me the successes I have attained professionally, academically, and personally.

Table Of Contents

List of Figures

List of Tables

<u>**Chapter One**</u>

Thesis Introduction and Problem Statement

1.1 Introduction

A poorly designed rigging system can produce cold shuts, inclusion defects, gross shrinkage, and porosity in a casting. Shrinkage and porosity defects occur as a direct result of poor directional solidification. Accordingly, foundrymen continuously seek to improve the modeling of the solidification behavior of cast metals, and have developed a host of computer applications for this purpose. These programs include 2D and 3D finite difference or finite element heat transfer programs, which often incorporate fluid flow analysis, to provide a detailed picture of the cooling behavior of castings. These applications are often expensive to purchase, time consuming to run, and require extensive training to operate effectively. Therefore, while these applications can produce precise results, they are less than ideal for the foundry engineer seeking quick design iterations in order to optimize the rigging configuration.

Before design verification with a finite element/finite difference code, a quick, computerized check that expends very little CPU time, executes on a PC, and provides a reasonably reliable evaluation of the design would facilitate rapid design iterations. Several such programs exist, especially for 2D models. Most use Chvorinov's rule, which relates the casting modulus to its cooling time, in order to provide a useful prediction of the part's solidification behavior. This method is greatly aided by the use of feature based design, since a feature's data structure can incorporate the analysis information. However, a solid model generated using a different design application may not have the required feature library, which would force a remodeling of the part, or force the analysis program to segment the model into a compilation of features which approximate the geometry.

Another method involves using a greatest included sphere procedure to determine the solidification time of a part: it assumes that the larger the sphere included in a given section of a casting, the longer the time required to fully solidify that section. This procedure developed from the greatest included circle method, which is a 2D, hand-performed technique. The foundryman uses a circle template and a 2D print of the casting in order to determine the largest circles he can fit into the casting design at various locations on its geometry [ICI68]. These critical locations include thick regions of the casting and its paths for make up metal flow during solidification. The greatest included circle method assumes that the center of any included circle will take longer to solidify than any other point in the circle. Furthermore, the relative time required to solidify any given included circle is assumed to be directly related to the size of its radius. Therefore, the large, isolated circle of Figure 1.1 indicates that its center will freeze after the centers of the two circles that flank it. Since the two flanking circles are located in the paths of the only two sources of makeup metal for the thick center region, the greatest included circle predicts that shrinkage will result for this geometry.

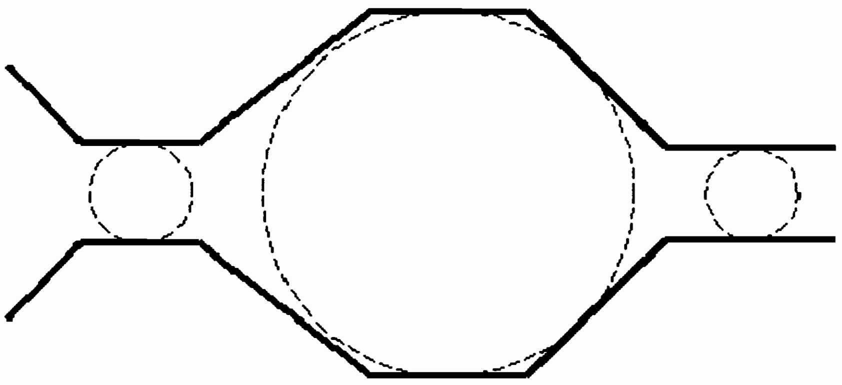


Figure 1.1: Casting cross-section with greatest included circle shown. The isolated large circle in the middle indicates that shrinkage could result do to a lack of feed metal during the cooling process.

By extending the greatest included circle method to three dimensions and automating the method, a computer program can evaluate the solidification behavior of a non-feature based solid model. However, unlike the foundryman, a computer cannot use personal experience to identify the regions of concern for a given casting geometry. In other words, the computer cannot readily recognize thick and thin regions of castings, since these terms are qualitative. The medial surface transformation, otherwise known as a medial axis transformation (MAT) of a three dimensional object, resolves this problem by providing a mathematical method for identifying regions of concern in a casting solid model. The medial axis transformation is defined as the set of the centers of all the locally maximal included circles in a two dimensional object [Preparata77]. The medial surface transformation is therefore the set of the centers of all the locally maximal included spheres within a three dimensional object. The following paragraphs explain how the medial surface transformation of a casting solid model can be used to approximate its solidification pattern. They also introduce the Voronoi diagram, which is used to construct the medial surface transformation.

In order to understand the intuitive relationship between the medial surface transformation and a casting's solidification pattern, one may consider a semi-infinite solid slab undergoing cooling via conduction [Figure 1.2]. The cooling front in this simple one-dimensional case is a line parallel to the edge of the slab boundary and proceeds in a direction normal to the boundary. This concept extends to three dimensions as follows: the boundaries of an object undergoing cooling are approximated by planar boundary elements, called facets [Figure 1.3]. Since each facet separates the hot object from its cool surroundings, a cooling wave front initiates from each facet and extends into the object in directions normal to its facet of origin.

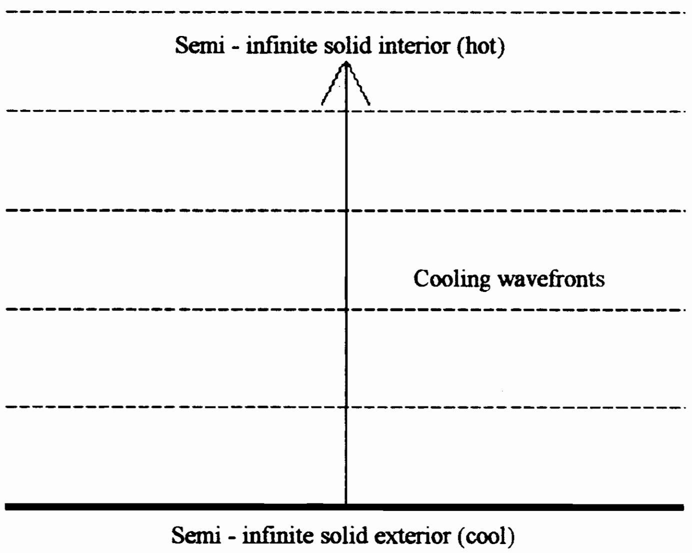

Semi - infinite solid interior (hot)
Cooling wavefronts
Semi - infinite solid exterior (cool)

Figure 1.2: Semi - infinite solid undergoing cooling with a uniform temperature boundary condition.

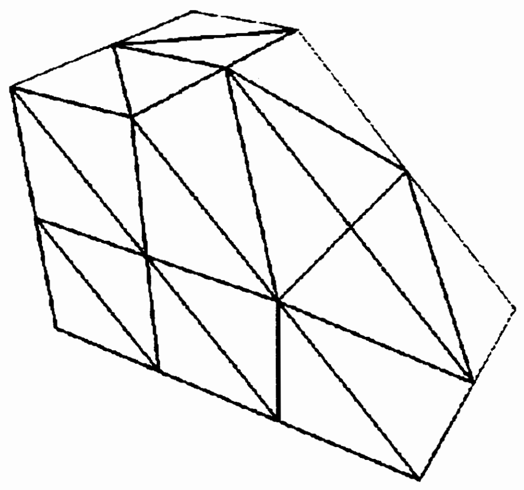


Figure 1.3: 48-facet solid model of a wedge.

The greatest included sphere method assumes that the heat transfer model described above adequately approximates the actual cooling process. Curves and surfaces formed by the intersection of any two cooling wave fronts comprise the set of points which are an equivalent distance to at least two of the model's facets. Since the medial surface transformation of the faceted model also comprises of this same set of points, it can be used to find the hot spot of the model by determining which of these points is furthest away from its closest facets.

The Voronoi diagram presents a straightforward method for calculating the location of these wave front intersections [Figure 1.4]. Dobkin describes the Voronoi diagram as defining "for each site the region of space for which it is the closest site" [Dobkin92]. A site can be any geometric entity from which a distance can be measured to any arbitrary point in the space. With respect to this thesis, the site is defined as a planar facet of a casting model. The Voronoi diagram is useful because the medial surface transformation of a polyhedral object is a subset of the Voronoi diagram of that object. In the case of convex solids, the Voronoi diagram of the interior of the solid is equivalent to the solid's medial surface transformation. Since this thesis restricts itself to convex geometries, the terms Voronoi diagram and medial surface transformation will be used interchangeably throughout this thesis in order to describe the medial surface transformation. In particular, the name Voronoi diagram will be used to describe the actual transformation as implemented in the thesis code.

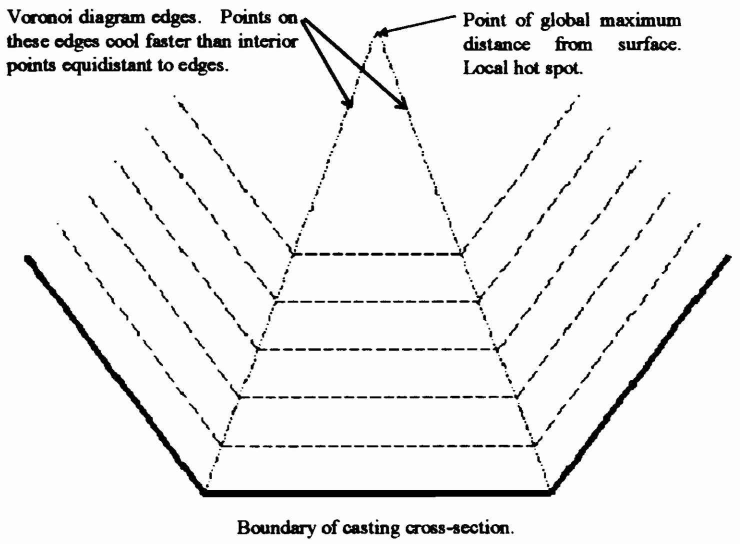

Voronoi diagram edges. Points on these edges cool faster than interior points equidistant to edges.
Point of global maximum distance from surface. Local hot spot.
Boundary of casting cross-section.

Figure 1.4: Partial cross-section of a casting undergoing uniform cooling. The bisecting lines above are a subset of the Voronoi diagram of the three boundary edges shown. Points along the Voronoi edges will cool faster than those points not on the diagram; however, the cooling pattern will converge to the globally maximal point at the intersection of the two Voronoi edges. Therefore, this point is a hot spot.

The Voronoi diagram provides an added benefit: it is an excellent starting point for comprehensive finite element analysis. It is the dual of the Delaunay tetrahedralization and may therefore be used to generate a well-conditioned finite element mesh of the casting. The mesh's good conditioning arises from two properties of Delaunay tetrahedralizations. By definition, Delaunay tetrahedralizations maximize the minimal angle for all tetrahedrons in the mesh [Dobkin92], which minimizes element distortion. In addition, their relationship to Voronoi diagrams ensures that they are optimally oriented with respect to the flow of heat within the casting.

The medial surface transformation of an object also provides an added benefit: it is well suited for implementation as an input data set for automated casting design applications. The transform provides a compressed, unique representation of an object, from which the original object's geometry can be retrieved [Sudhalkar93]. Therefore, the medial surface transformation of a casting can provide a quantitative approximation of its cooling pattern embedded in a data set which compactly and completely describes the casting geometry.

Voronoi diagrams and medial axis/surface transformations (MATs) are well understood and extensively utilized geometric entities; their applications range from oil well reservoir modeling and analysis [Palagi93] to fractal generation [Shirriff93]. From the above discussion, it is clear that they also present a straightforward means of automating the greatest included sphere method for approximating casting solidification. This thesis investigates whether such an automated greatest included sphere method can rapidly approximate solidification patterns while still providing acceptable accuracy for performing design iterations.

1.2 Formal Problem Statement

This thesis seeks to demonstrate the feasibility of using a medial surface transformation to implement a rapid, automated greatest included sphere method for approximating the solidification pattern of faceted models of castings. To maximize the application's availability, it accepts CAD models described in the .STL file format, which is the rapid prototyping industry's de facto standard CAD model file format [Bøhn93]. From this file format, the Voronoi diagram, which is equivalent to the medial surface transformation in the case of convex faceted models, will be computed. The generation of the Voronoi diagram, and the construction of the MAT from the Voronoi diagram, for general faceted models is left for further research.

1.3 Solution Overview

The experiment of using a Voronoi diagram/MAT based approach for rapid solidification modeling approximation consists of three separate stages. First, the computer constructs the Voronoi diagram of a faceted solid model. Then, a rendering program provides for post processing and visualizing the Voronoi diagram. Finally, an additional program verifies the solidification pattern predicted by the Voronoi diagram through comparing data points on the Voronoi diagram with corresponding nodes from an explicit FDM model of identical geometry.

The Voronoi diagram generation algorithm computes the Voronoi diagram of a faceted solid model by partitioning the interior points of the model according to which facet they are nearest. For a convex faceted model, a single plane partitions the interior points which are closest to one of any pair of the model's facets. If these partitioning planes, or bisecting planes, are found between a given facet and each of the other facets in the model, then the set of points which are strictly closest to that facet can be identified by intersecting the set of the facet's bisecting planes. The algorithm performs this operation for each facet in the model. It first creates a polyhedron from the facet and the intersection of three of its bisecting planes. Then, it sequentially intersects each of the remaining bisecting planes with the polyhedron. These sequential intersections trim away more and more of the original polyhedron, until the polyhedron that remains contains only those points which are strictly closest to the facet. This process is repeated for each facet in the solid model, until the entire solid model interior is partitioned.

The interior points of the solid model are partitioned by planar faces, which are bounded by linear edges, which are in turn bounded by their endpoints, or vertices. A rendering program shows the relative distances of each of these entities to the solid model surface by computing the relative distance to each vertex on the Voronoi diagram, and rendering the faces, edges, and vertices using a color scheme based upon a linear interpolation of the vertices' relative distances. The vertices which are a maximum relative distance away from the model surface are rendered in pure red; those which coincide with the model surface are rendered in pure yellow. Intermediate vertices have a corresponding shade of orange. The colors of the edges and faces are rendered by linearly interpolating the colors of each of their vertices in order to determine the appropriate color for each interior point. The hot spot of the casting appears as a pure red point, edge, or face.

Once the Voronoi diagram is generated and the relative distances of its vertices are used to approximate the solidification pattern, the solidification pattern is verified through a comparison between the relative distances of the Voronoi vertices and the relative temperatures of nodes of an explicit finite difference method (FDM) model of the casting. Additionally, implementation issues such as robustness and run time are investigated.

1.4 Thesis Organization

This thesis consists of six chapters, including this introductory chapter. Chapter Two provides the mathematical background required to understand Voronoi diagrams and how they are generated for 3D convex faceted solid models. It also describes the heat transfer concepts required to understand the finite difference method used to demonstrate the feasibility of using Voronoi diagrams and MATs in solidification modeling. Chapter Three provides a literature review which discusses Voronoi diagrams, MATs, and their engineering applications, reviews current solidification modeling approaches, and discusses the issues involved in creating automated foundry design applications. Chapter Four discusses the programming and experimental methods used to create and test the Voronoi diagram model. Chapter Five provides the experimental results associated with generating the Voronoi diagram, compares it with an explicit FDM model, and discusses its computational and algorithmic complexity. Chapter Six reviews the major conclusions of this investigation and offers suggestions for further study.

Chapter Two

Mathematical and Heat Transfer Background

This chapter provides the information required to understand the algorithms used to construct the Voronoi diagram of a convex faceted solid model and the heat transfer techniques used to verify that the Voronoi diagram can accurately model cooling patterns in castings. The mathematics portion of this chapter defines the geometric entities used in this thesis and their corresponding data structures, and describes some basic algorithms required to generate a Voronoi diagram of a convex polyhedron. The heat transfer portion of this chapter discusses the physical concepts required to understand the cooling process this project models, and the limitations inherent in the model.

2.1 Mathematical Background

This section concerns the required geometry and data structures required to implement a Voronoi diagram generation algorithm. The first two subsections define the Voronoi diagram, the medial axis transformation, and their generalizations. The third subsection defines the geometric entities used as a basis to construct the Voronoi diagram of a convex faceted solid model, along with their program specific data structures. The final subsection discusses the basic algorithms required to transform a convex, faceted model into a Voronoi diagram of planar faces.

2.1.1 The Voronoi Diagram

The classical Voronoi diagram partitions a plane into regions which contain all the points closest to specified points, which are called generators or sites. Its applications range from optimizing neural networks to choosing the best location for a fast food restaurant based upon the locations of existing fast food restaurants [O'Rourke94]. Voronoi diagrams can be understood by considering a forest fire analogy. If a number of lightning bolts ignite several trees in a forest simultaneously, the individual fires spread from their origins in all directions, consuming all of the available trees in their path. When the flames of two separate fire fronts meet, no more trees remain to burn, so the fronts extinguish each other. The lines where each fire front meets another form a Voronoi diagram of the forest. Each lightning-struck tree is a site.

O'Rourke [O'Rourke94] defines the Voronoi diagram and Voronoi territory mathematically as follows: a set of sites, $P = \{p_1, \ldots, p_n\}$ is chosen in the 2D Euclidean plane. The Voronoi territory, $V(p_i)$, of the site p_i is defined as the partitioning of the plane which contains all points which are at least as close to p_i as to any other site. O'Rourke expresses this definition of a Voronoi territory mathematically as

$$V(p_i) = \{x : |p_i - x| \le |p_j - x|, \forall j \neq i\}. \qquad (2.1)$$

The Voronoi diagram is the set of all points which do not belong to a single, unique Voronoi region. This set corresponds to the union of the edges of all Voronoi regions. Voronoi territories are also known as Voronoi regions, Voronoi polygons, etc. This thesis refers to these entitites as Voronoi territories throughout.

Preparata and Shamos [Preparata85] take a halfplane approach to the definition of the Voronoi diagram. Taking the same set of sites *P* as that described above, they define $H(p_i,p_j)$ as the halfplane of p_i and p_j which is formed by the perpendicular bisector of line segment $\overline{p_i p_j}$. A halfplane is a line which divides a plane into two distinct subplanes. A Voronoi territory can then be expressed mathematically as

$$V(p) = \bigcap H(p_i, p_j), \forall i \neq j . \tag{2.2}$$

Note that this territory is the boundary which partitions the points whose nearest site is p_i, not the locus of points itself as O'Rourke [O'Rourke94] defines. In other words, the Voronoi diagram can be considered to be the union of the set of its territories, or the union of the boundaries of those territories, as is convenient.

Preparata and Shamos [Preparata85] also discuss generalized planar Voronoi diagrams. To generalize the Voronoi diagram of a set of sites *P*, each site is defined as a cluster of two or more points rather than a single point. For example, the Voronoi territory of a pair of points is the locus of points in the plane which are closer to that pair of points than to any other pair of points. Voronoi diagrams can also be generalized to higher dimensions; rather than restricting *P* to two dimensions, the set of points can exist in a Euclidean space of any dimension [Dobkin92]. In the case of three dimensions, the

Voronoi diagram generalizes to 2D boundaries, called faces, which describe a set of 3D territories.

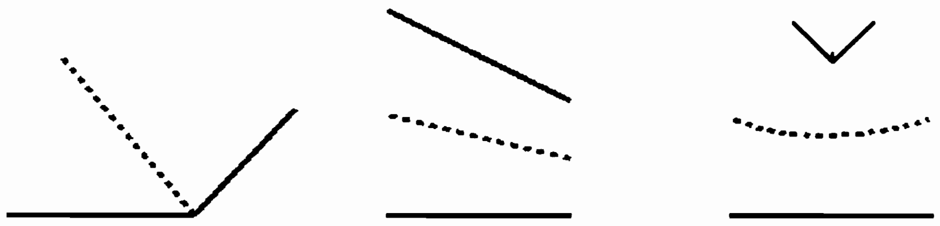
The Voronoi diagram of a convex, triangular faceted, solid model represents a subset of 3D generalization. Each site is a triangular facet, which is a continuous subplane of points, and the diagram itself is 3D. By restricting the sites to triangular facets, and by considering only the subset of the Voronoi diagram that lies within the boundaries of the solid model, the resultant Voronoi diagram is the union of the territory boundaries generated for every facet. These territories are all convex polyhedrons, and their boundaries are all planar faces. Construction of the Voronoi diagram therefore simplifies to an iterative series of plane/polyhedron intersections.

The Voronoi diagram provides an intermediate data structure from which the medial axis transformation may be easily derived. Generally, the complete medial axis transformation is a subset of the Voronoi diagram. In the specific case of convex faceted models, the medial surface transformation (which is a 3D generalization of a medial axis transformation), is equivalent to the Voronoi diagram on the interior of the model [Lee82]. This equivalency allows the medial surface to be found by using the straightforward approach above to calculate the Voronoi diagram, and then using the Voronoi diagram as the medial surface of the model. The following subsection provides an overview of the basic medial axis transformation of a polygon, and then discusses how it can be generalized to handle faceted solid models.

2.1.2 The Medial Axis Transformation

The medial axis transformation (MAT) of a polygon is closely related to the Voronoi diagram of a planar set of points. Just as the Voronoi diagram may be visualized using a forest fire analogy, the MAT can be visualized using a prairie fire analogy [Blum67]. If an isolated prairie field is set ablaze at all points along its boundary, the fire front will proceed towards the interior of the field, consuming all of the grass in its path. Whenever two opposing fire fronts meet, they will extinguish each other. The lines or curves at which these fire fronts meet form the medial axis of the field's polygonal boundary. Thus, a MAT is a subset of the generalized Voronoi diagram in which the sites are the edges of a polygon [O'Rourke94].

A medial axis transformation of an arbitrary simple polygon, *G*, is a set of points internal to *G* which have more than one closest point on the boundary of *G* [Preparata77]. If the polygon is convex, the MAT will comprise entirely of line segments; if the polygon is non-convex, the MAT will comprise of both line segments and parabolic segments. Linear medial axis segments arise from bisecting the convex angles formed between each convex pair of edges in the polygon. Parabolic segments arise from mapping the points which are closest to both a reflex vertex of a polygon and a facing edge of the polygon [Figure 2.1a-c].



(a): Medial axis segment of adjacent edges.

(b): Medial axis segment of non-adjacent edges.

(c): (Parabolic) medial axis segment of a reflex vertex and an edge.

Figure 2.1: Medial segments of various entities of a polygon.

As mentioned in Section 2.1.1, Lee [Lee82] points out that the medial axis of a simple polygon is a subset of the generalized Voronoi diagram of the polygon. Specifically, the medial axis segments can be extracted from the Voronoi diagram of the polygon by removing the Voronoi edges which are incident with each reflex vertex. Since a convex polygon has no reflex edges, the medial axis is equivalent to subset of the polygon's Voronoi diagram which lies within the polygon's interior.

The medial axis transformation of a polygon can be generalized for polyhedrons by first computing the generalized Voronoi diagram of the polyhedron, and then extracting the MAT from the Voronoi diagram. General polyhedral solids may have a variety of reflex edge and reflex vertex configurations, such as saddle points, peak points and peak edges. The resulting non-planar faces which form boundaries for the Voronoi territories of these entities are more complex to compute. Therefore, this thesis restricts itself to convex polyhedrons in an attempt to focus on the feasibility of using the Voronoi diagram, which is equivalent to the MAT for convex polyhedrons, in approximating solidification in castings.

With convexity enforced, the straightforward computation of the generalized Voronoi diagram of a 3D convex faceted solid model can be used directly to compute the generalized MAT of a 3D convex faceted solid model. This interchangability means that although the greatest included spheres solidification modeling method arises from the definition of the medial surface transformation, all of the special properties of the Voronoi diagram may also be applied to the casting model if desired. For example, as mentioned in Chapter One, the Voronoi diagram can be used for optimal finite element mesh generation. Although the strict equivalence between these two data structures is lost for general solid models, they remain closely enough related that one may trivially be constructed from another.

2.1.3 Geometric Entities

This section reviews the basic geometric entities required to understand the construction of Voronoi diagrams for convex faceted solid models and provides a description of the data structures used to represent these entities in the program developed for this thesis. These entities include the polyhedron and its boundary elements, the faceted solid model, and the Voronoi territory and its components.

2.1.3.1 Basic Geometric and Topologic Definitions

Polyhedron A solid, three dimensional entity which can be defined in one of two ways: either as a region of space whose boundary comprises a finite number of faces, any pair of which are either disjoint or meet at edges and vertices (topologic definition) [O'Rourke94], or as a region of space which is bounded by simply connected, intersecting facets (geometric definition). This thesis uses both definitions. The appropriate definition for a given situation can be determined from context.

Point A zero dimensional geometric entity which defines a location in space.

Vertex A zero dimensional topological entity [O'Rourke94].

Line segment A one dimensional geometric entity which consists of the set of points located between two endpoints.

Edge A one dimensional topological entity which connects two adjacent vertices [O'Rourke94].

Facet A simple closed two dimensional geometric entity, or polygon. In this thesis, it consists of three edges and a directed normal which defines the material side of the facet.

Face A two dimensional polygon on the boundary of a polyhedron [O'Rourke94]. The face of a polyhedron is in turn bounded by a closed set of edges, which connect a chain of vertices around the border of the face.

Halfspace A set of points on or to one side of a plane [O'Rourke94]. A half space can be defined by a single point on its boundary plane and a directed vector normal to the boundary plane of the halfspace. A polyhedron may be represented as an intersection of four or more halfspaces whose planar boundaries coincide with the faces of the polyhedron, and whose direction vectors point to the exterior of the polyhedron.

Bisecting Plane This entity is specific to the program used in this thesis: it is a halfspace whose bounding plane exactly partitions the intersection of a pair of halfspaces into two equally sized subspaces. It is called a bisecting plane because its normal bisects the angle between the normals of the bounding halfspaces. Since bisecting planes are themselves halfspaces, they have a directed normal which identifies one of the two subspace partitions as positive, and one as negative. If the pair of intersecting halfspaces are faces of a polyhedron, the bisecting plane defines which portion of the polygon's interior is closer to one face than the other.

2.1.3.2 Data Structure Implementations of Basic Geometric Entities

The program used to calculate the Voronoi diagram and medial surface transformation of a .STL solid model implements two basic data structures: a faceted solid model, and the Voronoi territory belonging to each of its facets. Both of these data structures represent polyhedral solids, which are each customized as necessary.

Faceted Solid Model. The model is an array of facets. These facets are records which consist of the following data: an array of pointers to each of its three vertices, a pointer to its normal vector, an array of pointers to each of its three adjacent facets, a double precision plane constant, and an integer enumerating the territory type. The facet can be manipulated as a halfspace by evaluating

$$Ax + By + Cz = D \tag{2.3}$$

where *A, B,* and *C* represent the coefficients of the directed normal, and *D* is the facet's plane constant. By convention, the normal points toward the exterior of the solid. The territory type enumeration specifies the structure of the initial Voronoi territory formed by using the facet and its three adjacent facets as sites. When these four sites are coplanar, the facet's territory is a triangular prism; otherwise, the facet's territory is a tetrahedron, or pyramid, whose apex is the point of intersection between the three bisecting planes created between the facet and each of its three adjacent facets. The program creates these four site initial Voronoi territories as a first step in defining the completed Voronoi diagram.

Voronoi Territory. Like the faceted solid model, the Voronoi territory is a supplemented list of its faces. The face list is doubly linked and circular, which maximizes programming flexibility. Each territory also references its parent facet, and references a list of one or three peak points. The peak points are either the set of three points created for a prism type initial territory, or a single point corresponding to the apex of a pyramid type initial territory. For this thesis, the geometric entity used as a site to generate the Voronoi territory is a triangular facet of the faceted solid model.

Territory Face. The territory face is another supplemented list of its boundary elements; in this case, a doubly linked, circular list of its edges. It contains a reference to a normal vector, a double precision plane constant, and an integer visit flag. Like a facet, a territory face can also be manipulated as a halfspace by using its normal and plane constant in Equation 2.3. Since Voronoi territory construction is an iterative series of plane/polyhedron intersections, the visit flag is used to show whether a face has been visited by the intersection marching algorithm during the current intersection. The marching algorithm is described in the following section and in Chapter Four.

__Territory Edge.__ The edge provides the connectivity information required to navigate a Voronoi territory. As implemented, its data structure closely resembles the winged-edge data structure [O'Rourke95], which is a classic structure for representing polyhedra. A winged edge consists of references to two vertices, a left and right side, the faces lying to its left and right, and to the four edges which are adjacent to one of the edges vertices and one of its faces [Figure 2.2].

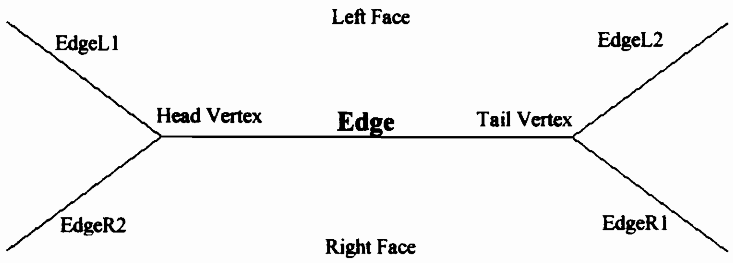


Figure 2.2: Winged edge data structure.

This thesis implements an edge data structure which references a head vertex, a tail vertex, a parent face, an adjacent face, the next and previous edges on the parent face's boundary, and a twin edge, which is a duplicate edge whose parent face is the edge's adjacent face. The orientation of each edge is implied by the position of its head and tail vertices, and by the next and previous edges in the face's list.

2.1.4 Geometry-Based Algorithms and Procedures

This section discusses the basic geometric and topological concepts required to complete the construction of the Voronoi diagram. Plane/plane intersections comprise the first topic of discussion; they are required for computing the halfspace equations for Voronoi faces. Line segment/plane intersections, which are required for computing the vertices that define a Voronoi face's boundaries, comprise the second topic. Intersection marching algorithms comprise the third.

2.1.4.1 Plane/plane intersections

The bisecting plane between a pair of facets is created by intersecting the halfspaces of two facets. Since a halfspace is defined by a normal vector and a point, one of each must be calculated for each bisecting plane.

The normal vector of the bisecting plane is found by performing a vector addition upon the unit normal vectors of the intersecting facets [Figure 2.3]. The resultant vector is the bisecting plane normal because it lies in the same plane as the two original normals and bisects their included angle. The latter fact is true since the dot product between the normal, n_1, of intersecting plane, $\mathbf{P_1}$, and the resultant vector equals the dot product of the normal, n_2, of intersecting plane, $\mathbf{P_2}$, and the resultant vector:

$$\begin{aligned} n_1 \cdot (n_1 + n_2) = n_1 \cdot n_1 + n_1 \cdot n_2 = 1 + n_1 \cdot n_2 \\ n_2 \cdot (n_1 + n_2) = n_2 \cdot n_2 + n_2 \cdot n_1 = 1 + n_2 \cdot n_1 \end{aligned} \tag{2.4}$$

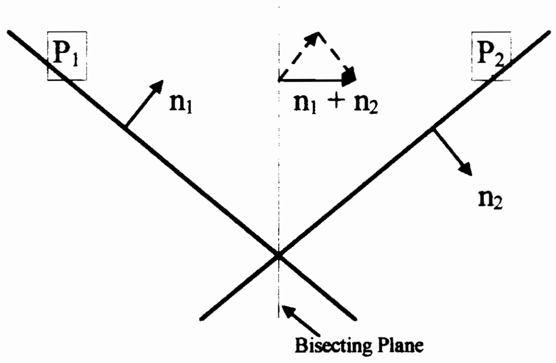


Figure 2.3: Bisecting plane of planes $\mathbf{P}_1$ and $\mathbf{P}_2$. Its normal vector is found by adding $\mathbf{n}_1$ and $\mathbf{n}_2$.

Once the bisecting plane normal is found, its halfspace is fully defined by ensuring that it passes through a point common to both of the original facet halfspaces. This point can be found by intersecting $\mathbf{P}_1$ and $\mathbf{P}_2$:

$$\begin{aligned} A_1x + B_1y + C_1z &= D_1 \\ A_2x + B_2y + C_2z &= D_2 \end{aligned} \qquad (2.5)$$

where $n_1 = [A_1,B_1,C_1]$, $n_2 = [A_2,B_2,C_2]$, and D_1 and D_2 are plane constants for $\mathbf{P}_1$ and $\mathbf{P}_2$. This system of equations defines the intersection of the planes $\mathbf{P}_1$ and $\mathbf{P}_2$. The system yields a line of intersection. A point along this line can be found by setting by either *x, y,* or *z* to zero.

Bisecting planes must also be defined for pairs of opposite facing parallel faces, for example the opposite faces of a cube. The bisecting plane in this case is the midplane of the pair. The normal of the midplane is defined by assigning it the common normal of the parallel faces. The point used to complete the midplane's definition is found by arithmetically averaging the coordinates of any point on $\mathbf{P}_1$ with any point on $\mathbf{P}_2$ [Figure 2.4].

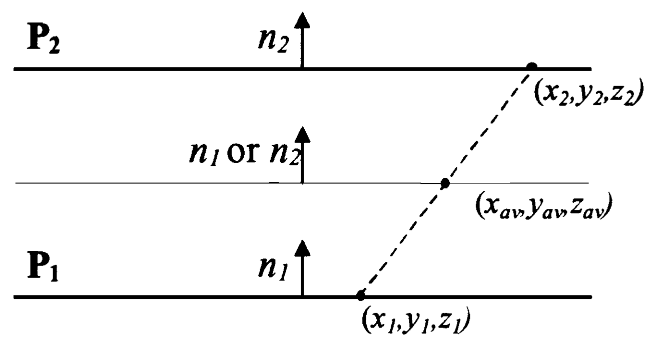


Figure 2.4: Construction of midplane for parallel planes P_1 and P_2.

2.1.4.2 Line segment/plane intersections

In order to insert a new Voronoi face into a territory during its construction, the new face's plane must be intersected with the existing territory. The points used to define the edge vertices of the new face are calculated by intersecting the line segments which define the edges of the existing Voronoi faces with the plane upon which the new face lies.

The process consists of two steps: 1) identify an intersecting line segment, and 2) calculate the point of intersection between the line segment and the plane. Intersecting line segments are found by substituting the coordinates of each of its endpoints into the equation of the halfspace upon which the new face lies. Because the halfspace has a defined orientation, the results of these substitutions will have opposite signs whenever the line segment's two endpoints fall on opposite sides of the halfspace. Once an intersecting line segment is identified, the point of intersection is found by computing the parametric distance from one of the endpoints to the plane, along the line segment's direction vector. The parametric distance, t, is defined as:

$$t = \frac{x - x_o}{V_i} = \frac{y - y_o}{V_j} = \frac{z - z_o}{V_k}, \tag{2.6}$$

where $P = (x_o, y_o, z_o)$ is an endpoint, and $V = [V_i, V_j, V_k]$ the line segment's direction vector. Given the halfspace equation of the new face, t becomes

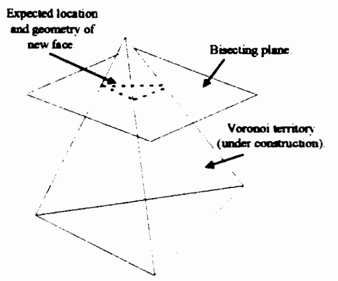
$$t = D - \frac{n \cdot P}{n \cdot V} \tag{2.7}$$

where D is the halfspace plane constant and n is the halfspace normal. Once t is found, the intersection point is solved for using Equation 2.6.

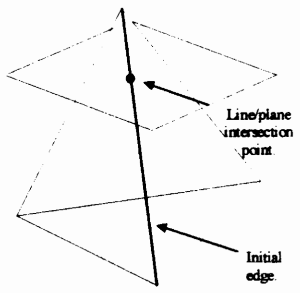
2.1.4.3 Intersection Marching Algorithms: Polyhedron/Halfspace Intersections

An intersection marching algorithm is used to construct the Voronoi territory of a facet by incrementally updating the facet's initial Voronoi territory through the intersection of the territory with the facet's bisecting planes [Figure 2.5a]. The intersection marching algorithm exploits the topology of the polyhedron in order to minimize the computations required for polyhedron/halfspace intersections, ensures that the topology of the resulting polyhedron is preserved, and guarantees that the polyhedron remains closed and connected.

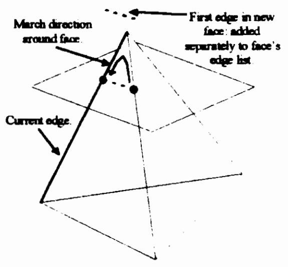
The algorithm begins by naively searching the faces of the polyhedron for an edge whose corresponding line segment intersects the halfspace, and finds its corresponding point of intersection [Figure 2.5b]. Once the initial edge is found, the algorithm uses edge/edge connectivity to march through the initial face's edges to find a second edge whose corresponding line segment intersects the halfspace, and computes its point of intersection. Once a second edge is found, the new face's first edge is created by creating vertices which correspond to the two points of intersection, and connecting those vertices [Figure 2.5c]. The new edge is inserted into the new face's edge list, and the algorithm proceeds from the current face to the adjacent face of the second edge [Figure 2.5d]. The marching algorithm finds the second intersecting edge on the second face. This process continues until the initial edge is rediscovered; the new face's final edge is created by connecting the second vertex of the previous edge in its list with the first vertex of the initial edge [Figure 2.5e]. Once the algorithm returns to the initial edge, it closes the new face's edge list, and inserts the new face into the territory's face list. Its adjacent faces are trimmed accordingly [Figure 2.5f].


Expected location and geometry of new face
Bisecting plane
Voronoi territory (under construction)

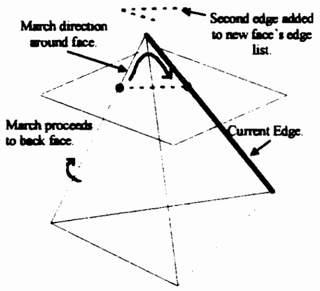
a. Intersection of a bisecting plane and an under construction Voronoi territory.



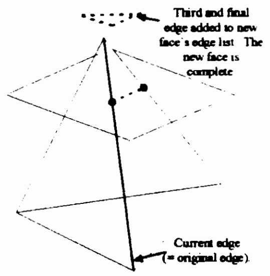
b. First step: an initial edge is found, and a point of intersection between the edge's line segment and the bisecting plane is calculated.


March direction around face
First edge in new face: added separately to face's edge list
Current edge

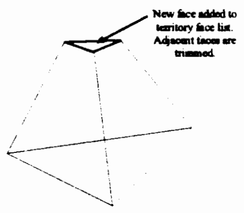
c. Algorithm marches around the current face's edge list until second intersecting edge is found. A new edge is created and added to the new face's edge list.


March direction around face
Second edge added to new face's edge list
March proceeds to back face
Current Edge

d. The march proceeds from the first face to the back face, and then around the back face until an intersecting edge is found. A second edge is created and added to the new face.


Third and final edge added to new face's edge list. The new face is complete
Current edge (= original edge)

e. Algorithm repeats until the original edge is found. The new face is now closed.


New face added to territory face list. Adjacent faces are trimmed.

f. Once the new face is complete, it is incorporated into the territory face list.

Figure 2.5: Intersection marching algorithm.

Traversing the polyhedron in this manner forces the new face to have a fully closed boundary of edges. Should any error occur during the intersection march, the algorithm can abort the march since the new face is constructed as an independent entity. This enhances the robustness of the algorithm. Additionally, by keeping a history of the new face's creation, the intersecting faces of the polyhedron are efficiently trimmed, and their connectivity with the new face is guaranteed. Faces whose facets do not intersect the new face's halfspace are guaranteed to lie either above or below the new face; therefore, once the march is completed, an existing face which lies completely outside the new boundary of the polyhedron can be detected by inserting a single vertex of that face into the new face's halfspace equation.

2.2 Heat Transfer Background

This section provides the basic heat transfer knowledge required to compare the greatest included sphere method with an explicit finite difference method for approximating heat transfer in a casting. The first subsection discusses the explicit finite difference method, which will be used as the benchmark for comparison with the Voronoi diagram based implementation of the greatest included sphere method of solidification modeling. The second subsection discusses Chvorinov's Rule, a classic, time tested geometric method for modeling casting solidification, and compares and contrasts Chvorinov's rule with the greatest included sphere method. The third subsection lists the assumptions and limitations applicable to using Voronoi diagrams to automate the greatest included sphere method.

2.2.1 The Explicit Finite Difference Method

The finite difference method (FDM) is an iterative numerical method for solving transient and steady state problems in heat transfer. The method approximates the geometry of the object undergoing heat transfer by applying a grid of regular elements to the object. Once this grid is established, nodes are created which approximate the volume of each object as a concentrated point location. The user specifies the initial heat transfer conditions of each node, a set of material heat transfer properties, boundary conditions for the model, and the location of heat sources and sinks. Once these preprocessing steps are completed, the finite difference algorithm computes the temperature at a given node by performing an energy balance which considers the heat transfer between that node and each of its surrounding neighbors [Kreith80].

Two basic forms of the finite difference method exist. Both involve using a regular grid to approximate the geometry and heat transfer characteristics of the model. The explicit finite difference method uses the temperatures of all the nodes at a current time step to compute the node temperatures at the next time step. Each node equation contains only one unknown, namely the temperature of that node during the next time step. The implicit method involves solving the future temperatures of all the nodes simultaneously. The major advantage of the explicit method is that it is simple to formulate, and requires less computational time per iteration. The major advantage of the implicit method is that it is stable, regardless of the time step chosen. Thus, it is more robust than the explicit method, which become unstable for excessively large time steps. The choice of which technique to use depends upon the geometry and boundary conditions for the model to be solved. Combinations of the two methods may provide the best approach to some problems [Kreith80]. This thesis uses the explicit finite difference method to generate comparison models for the greatest included sphere method.

The three dimensional energy balance used to describe the heat transfer of a given brick node 0 can be expressed in the following equation:

$$\sum_{i=1}^{6} kA_i \frac{T_i^t - T_o^t}{\Delta x_i} = \rho Vc \frac{T_o^{t+\Delta t} - T_o^t}{\Delta t} \qquad (2.8)$$

where k represents the conductance of the element material, A_i represents the area of the element's shared face with element i, T represents a node temperature, ρV represents the element mass, c represents the specific heat of the element, Δx_i represents the distance between node 0 and node i, and Δt represents the time step [Kreith80]. When cubic nodes are used, the equation can be reduced and solved for the new T_o as follows:

$$T_o^{t+\Delta t} = T_o^t + C\left(T_{sum} - 6T_o^t\right) \qquad (2.9)$$

where C is a general constant lumping the time step, geometry and heat transfer terms together, and T_{sum} represents the sum of the temperatures of the neighboring six finite difference nodes.

2.2.2 Chvorinov's Rule

Chvorinov's rule relates the solidification time of a solid to its geometry [Chvorinov40]. It is one of the oldest and most time tested of the geometric methods used to understand casting solidification [Upadhya93][DeKalb87][Neises87]. The rule states that, within the limitations discussed below, the time required to solidify a casting of a given solid geometry is proportional to the square of the casting's modulus, or volume to surface area ratio:

$$t_f = C\left(\frac{V}{A}\right)^2 \tag{2.10}$$

In this equation, t_f indicates the casting's final solidification time, C represents a constant of proportionality, and $\frac{V}{A}$ represents the casting's modulus.

The above equation only works accurately for low eutectic range metals, is valid only without superheating, and is inaccurate for complex geometries [DeKalb87]. Low eutectic range metals are metals whose phase transformations do not require a significant amount of energy. This limitation is necessary, since the geometric relationship only applies provided that the cooling front proceeds at a more or less constant pace from the surface to the interior of the casting. The prohibition on superheating ensures that the metal has an essentially uniform initial temperature. The accuracy of Chvorinov's Rule decreases as the geometric complexity increases due to the fact that certain geometric configurations, for example internal corners, lead to areas of concentrated heat transfer. This concept is analogous to stress concentrations arising from cracks in material under tension. Chvorinov's rule has been modified at the University of Wisconsin-Madison to allow general 2D cross-sections and long eutectic range alloys [DeKalb87][Neises87]. Their method also predicts localized heat transfer deviations due to internal and external corners.

2.2.3 Relating Chvorinov's Rule to the Voronoi Diagram Based Greatest Included Sphere Method

The greatest included sphere heuristic is similar in spirit to Chvorinov's rule. Both rely upon uniform cooling conditions at the boundaries, both assume a uniform initial temperature of the molten metal, and both relate geometry to solidification time. The greatest included sphere heuristic relates the radius of a sphere to the solidification time at the sphere's center. Chvorinov's rule states that a spherical casting's total solidification time is related to the sphere's modulus, which is in fact its radius.

Despite similar assumed conditions and a similar reliance upon geometry, however, Chvorinov's rule and the greatest included sphere method do not provide the same information. Chvorinov's rule provides a good estimate of total solidification time of a casting, but gives no information on local solidification times. The greatest included sphere method provides a rough, relative estimate of the solidification times between several points within a casting. Even though the radius of a sphere is also its casting modulus, the effective modulus of the greatest included sphere at an arbitrary point within a casting is dependent upon the casting's surface geometry. Therefore, no explicit physical relationship exists between the two methods. Lastly, Chvorinov's rule supplies exact solidification times for specific metals, while the greatest included sphere method only requires knowledge of the casting geometry.

Despite the differences between the two methods, several specific limitations of a Voronoi diagram implementation of the greatest included sphere method become apparent by comparing it with Chvorinov's rule:

1. Without subsequent modification, the Voronoi diagram will only produce good results for low eutectic range metals.
2. The Voronoi diagram will produce errors with respect to FDM/FEM models away from the boundaries of the Voronoi territories. This is because the diagram assumes that at least two cooling surfaces contribute significantly to the cooling of a given point. A point well inside of territory boundaries is dominated by only one cooling surface; therefore, it will cool at a significantly slower rate than a point on the Voronoi diagram whose distances to two or more casting surfaces are equivalent.

Since the greatest included sphere method relies upon symmetry to predict relative solidification times, it shares with Chvorinov's rule the assumption that all boundary conditions are uniform, and the at the interior's initial temperature is also uniform.

Chapter Three

Literature Review

This chapter discusses three major topics: current methods of solidification modeling for casting processes, Voronoi diagrams and medial axis transforms (MATs), and automated casting design systems. The solidification modeling subsection introduces the concepts behind solidification modeling methods, provides descriptions of both precise and approximate solidification modeling which are currently in use or under investigation, and lists the major commercial applications currently available. The discussion of medial axis transforms and Voronoi diagrams explains the methods used in their construction, outlines various types of transforms, and comments on the complexity of the algorithms used to generate them. It also describes existing applications for their use, and explains their potential application to solidification modeling and automated casting design. The automated casting design discussion explains the resistance of the foundry industry towards design automation software and describes the techniques used to create such software.

3.1 Solidification Modeling Review

Kannan et al. [Kannan90] state that the current thrust of solidification modeling research lies in the refinement of heat transfer and fluid flow techniques towards the goal of solving microstructural modeling problems. The present state of the art, which takes the form of 3D finite element and finite difference codes, fully coupled with a computational fluid dynamics simulation of the mold filling process, comes close to achieving this goal. However, it cannot completely address the microstructural formation of heterogeneous materials because of difficulties in modeling the random distribution of secondary inclusions in these materials. Ghosh et al. [Ghosh94][Ghosh95a][Ghosh95b], in combining Voronoi polygons and finite elements, achieve some success in modeling heterogeneous materials, but since much remains unknown about the process of nucleation in these materials, some level of uncertainty exists in their model. Brown and Spittle [Brown93] point out that while finite element and finite difference analysis methods can offer powerful solutions to solidification problems, they must have a physical model of the solidification process upon which to base their analysis. The physics of microstructural solidification is not well understood. In addition, the generation of accurate finite element and finite difference solutions may take prohibitive amounts of computational time.

While microstructural solidification is an important process to understand when optimizing casting material parameters, most casting applications require only an understanding of macrostructural solidification patterns in order to produce sound castings. Ideally, the engineer uses a fast, approximate method as an aid in making preliminary refinements to a design, followed by a full finite element or finite difference analysis to verify the design and optimize the material parameters. This review provides a survey of the methods used in performing solidification analysis on castings, starting with comprehensive finite element codes and concluding with rapid, approximate solidification modelers employing geometry based techniques. Since much research has already been performed and marketed, a summary of current commercial foundry packages is included.

Kannan et al. [Kannan90] list four methods of solidification simulation in their literature review: the integral profile method (IPM), the boundary element method (BEM), the finite difference method (FDM), and the finite element method (FEM). IPM uses a truncated power series approximation of a geometry's temperature profile to solve a heat transfer differential equation; this older method relies on the assumption of temperature independent thermophysical properties and is therefore rarely used. BEM discretizes the boundary of the geometry into elements, and then uses the inverse form of Galerkin's weighted residual method to solve an integral equation for each element. FDM converts the geometry into a regular grid of cells, and then uses iterative heat transfer equations to arrive at a solution. FEM reduces the geometry into polygons or polyhedra and uses the weak form of Galerkin's weighted residual method to process the model. Of these methods, FEM and FDM are by far the most widely used.

Of the solidification simulations available, FEM provides the most comprehensive geometric modeling capability of the physical processes involved. FEM thermal analysis can incorporate fully coupled fluid flow analysis in order to investigate the effects of solidification during and after mold filling. It has two advantages over FDM: it can consider heterogeneous materials [Kannan90] and it provides greater flexibility in modeling arbitrary geometries. However, FEM requires extensive computing power and knowledge of material properties in order to fully realize its potential. This review does not discuss FEM in depth, but rather shows a sufficient number of examples of solidification simulation via FEM in order to provide an understanding of its capabilities and limitations.

Tu et al. [Tu93] employ a typical, state of the art finite element application to investigate investment casting processes. It uses ProCAST (UES, Inc.), a fully coupled mold filling and thermal analysis code, to process a 35000 node, 25000 element model. The code accepts inputs quantifying the molten metal temperature, the mold preheat temperature, the metal inlet temperature and velocity, the metal and mold material properties, the effect of the insulation blanket and contact conductance between the mold and the metal, and radiation heat transfer boundary conditions. Output includes isochron plots, which show the time taken for a given location to freeze, temperature distribution vs. time plots, and plots showing the tendency for microporosity based upon one of two prediction criteria. Although it provides extensive information about the process, the extent of the preprocessing and analysis time required makes this approach unsuitable for conceptual design iterations.

Beffel et al. [Beffel89] also use finite element analysis to study an investment casting process. They use PATRAN/ANSYS to analyze a 9000 node, 5000 element model of investment cast turbine blade mold clusters. The application produces isochrons, grain size plots, cooling curves, and a plot of the tendency of an airfoil to undergo hot cracking relative to its length. Their work reveals the scope of information available from a typical FEM-based solidification simulator. It also reveals a need for considerable computing time; typical analysis runs lasted overnight on an FPS M-64/60 computer. While computing power has advanced considerably since the time of their report in 1989, it remains obvious that time is a distinct disadvantage in using FEM for rapid design iterations.

Kannan et al. [Kannan90] have developed a 2D finite element application called CADCAST, which combines mesh generation, material databasing, analysis, post processing and output plotting in a single, five module software package. The limitation to two dimensions indicates the willingness of the foundry industry to forego the accuracy of 3D codes in order to produce quicker solutions on less powerful computers.

While more limited in geometric modeling capabilities, the finite difference method can match the FEM in accuracy, and can effectively treat 3D casting models. The American Foundrymen Society has pursued research in FDM since the late 1960s [Pehlke88]. This research has resulted in two commercially available solidification simulators, the 2D AFSolid package and the 3D AFS Solidification System [Estrin94]. Their built in mold and material databases provide a crucial advantage, since FDM, like FEM, relies upon the quality of its materials property data, heat transfer coefficients, and its physical model of solidification in order to produce quality results [Brown93]. Each

FDM model may require complete reworking in order to accommodate the analysis of a new alloy.

Brown and Spittle [Brown93] avoid materials issues and realize an order of magnitude speed improvement over conventional FDM analysis with cellular automaton software. The procedure is described as similar to a time-stepping, dimensionless finite difference approach: it uses a 3D grid of finite difference cells and iterates temperature changes in each cell as a function of the cell's 6 nearest neighbors. It can treat chills, risers and cores accurately by representing them as separate boundary conditions, and it can predict macroshrinkage and macrofreezing patterns. Because it requires no specific information on the metal or mold type, it cannot directly predict freezing times. This drawback is inconsequential, however, to a foundry engineer interested in quickly locating regions in the casting requiring risers in order to prevent shrinkage defects.

Hill et al. [Hill91] also use a cellular method to arrive at a quick, approximate solidification time plot. They impose a uniform grid over the casting and perform a volume calculation which determines whether each cell is fully, partially, or not at all located within the boundaries of the casting. Then, they perform a weighted summation upon each grid point to determine its mass distribution. In this manner, the centers of the thickest sections, corresponding to the areas expected to solidify last, are identified along with the natural heat and feed metal flow paths into and out of those regions. The information gleaned from this analysis provides a rigging design expert system with a basis for design of risers and gating; the expert system can use the data from this analysis to quickly visualize the thick regions of a casting in the same way as a casting engineer does.

Of the geometry-based solidification modelers available, modulus-based methods based on Chvorinov's rule dominate. Chvorinov's rule relates the casting modulus, given by the ratio of the casting's volume to its surface area, to the time taken to freeze the casting [Upadhya93][Chvorinov40]. The mathematical expression for Chvorinov's rule is Equation 2.10. In this expression, C is a constant based on metal and mold material and temperature parameters. Chvorinov's rule was developed for short solidification range steels, but also applies to pure metals and low solidification range alloys, like those of aluminum and copper [DeKalb87]. Upadhya and Paul [Upadhya94] state that most applications utilizing Chvorinov's rule use some form of feature-based modeling or sectioning of a solid model in order to break a full casting geometry into simple components. While some of these applications are capable of performing 3D calculations, most only apply to 2D simplifications of 3D models. Computer programs employing this technique range in sophistication from those that require the user to perform sectioning by hand to those that compute a continuous distribution of modulus from a grid of points within a solid model.

Pei et al. [Pei87] present an application which forces the user to perform most of the work in processing a solid model for modulus calculations. The model is broken up by hand into basic shapes, such as spheres, cuboids, cylinders and frustums of cones. Where one shape connects to another, the overlapping surface area is recorded by hand as being round, cuboid, or cylindrical. With this information, the computer calculates section moduli for each component, and uses the numbers generated to compute riser dimensions.

Sirilertworakul et al. [Sirilertworakul93] section a casting geometry automatically using an AutoCad AME (release information not provided) function that sections a solid model and records the section's perimeter length and cross-sectional area. Since the sections are constrained to have uniform thickness, the modulus of each section can be calculated from these two parameters. Using this method, the program calculates the local modulus at several locations in the casting, and outputs either a color map showing the variation in modulus or a black and white grid of the casting which lists the local modulus of each grid square. Its main advantage is that it accurately predicts localized differences in cooling rates for internal and external corners.

Nieses et al. [Neises87] also compute the section moduli of 2D sections by using a generalized equation based upon an area/perimeter value for section modulus, similar to that used by Sirilertworakul et al. [Sirilertworakul93]. Nieses et al. also use a point modulus calculation which depends upon both the distance of a point to each edge of a section and a view factor for each edge with respect to that point. The view factor of an arbitrary internal point is related to the angle included between line segments drawn from that point to each of the vertices of a given edge.

DeKalb et al. [DeKalb87] combine this generalized section modulus formula with a considerable amount of experimental data to create an extension of Chvorinov's rule that accurately models long solidification range and eutectic alloys, such as high alloy steels, carbon steels, cast irons, aluminum alloys and copper alloys. Their application, called SWIFT, provides 41 different equations used to fit Chvorinov's rule to specific metals. This code can also calculate the start and end of the mushy zone of a long solidification range alloy at a given time. At the time of their report, SWIFT only applied to 2D models; however, the principle of section modulus calculations is extendible to 3D.

Upadhya and Paul [Upadhya93] use a distributed point modulus calculation to calculate the local section modulus of points in a finite difference mesh of a casting. The equation they use relates the modulus of a point to the summation of all the distances from that point to the mold boundary along a specified number of directions.

Many of the above solidification modeling concepts have been incorporated into commercial packages, among them SWIFT, MAVIS (cellular automaton), and ProCAST. Estrin [Estrin94] reviews 17 solidification packages from 13 companies, lists the required hardware platforms to run them, and describes the method they use to perform solidification simulations. Most of these applications can be run on PCs, and many operate on multiple platforms. There are ten FEM packages, of which seven have some fluid flow analysis capability. One is a combined FEM/FDM package, two are 3D FDM packages, and one is a 2D FDM package. All of the applications listed have either 3D capability or a companion package with 3D capability. Only one package, SWIFT, provides an exclusively geometry based modeler; two other applications, MAVIS/DIANA and RaPiDCAST, have both a quick geometry based modeler and a full FEM or FDM modeler.

Sandia National Laboratories' FASTCAST package may also soon become commercially available [Sandia92]. It provides multidimensional, nonlinear heat conduction analysis and radiative heat transfer analysis for investment casting simulation.

A Voronoi diagram based solidification modeling tool contributes to the existing body of casting solidification simulation applications by providing a fast approach to 3D solidification modeling which avoids the expense of full FEM/FDM applications. A greatest included sphere method based upon the Voronoi diagram is in philosophy most like Hill et al.'s [Hill91] mass distribution method. Both methods ignore localized boundary conditions such as chills and cores, rely exclusively upon the casting geometry to identify the thick portions of a casting, and provide information which can be used by a design automation expert system.

Although the accuracy of the Voronoi diagram based modeling method is not expected to be precise, its accuracy can be enhanced with a minimal speed increase by using a SWIFT style calculation on a gridded Voronoi diagram. The diagram requires gridding for this purpose because SWIFT is an iterative process which recalculates the local modulus of grid points based upon the solidification of the model up to a given time. A Voronoi diagram grid would be more sparse than the full FDM grid used by SWIFT, yet contain the locations of critical interest for heat transfer simulation. By combining these two ideas, the potential exists to achieve the accuracy of SWIFT with on a greatly reduced, yet much more physically significant, set of points than those created by naively gridding the full solid model.

3.2 Voronoi Diagrams and Medial Axis Transformations

An alternate definition of the 2D medial axis transformation (MAT, or skeleton) of an object is that it is the locus of the centers of all maximal disks in a 2D object [Rolland92]. Therefore, it provides the motivation for automating the greatest included circle method. The MAT generalizes in 3D to the locus of all maximal spheres in a 3D object, which provides the motivation for using the Voronoi diagram to model solidification by extending the greatest included circle method into an automated greatest included sphere method.

The medial axis transformation was first conceived as a tool used in pattern recognition and computational geometry to compactly represent and compute the geometric properties of digitized patterns in a grid [Preparata77]. Typically, computations are completed by discretizing the total area of the image into subareas of suitably small size, such as that of a picture element, and treating those subareas as a single point. However, exact algorithms exist for polygons. The following subsections will describe the methods used in MAT and Voronoi diagram computation and comment on their complexity, discuss the types of Voronoi diagrams and their potential applications, and relate this discussion to the application of MATs and Voronoi diagrams to solidification modeling and automated casting design.

3.2.1 Medial Axis/Surface Transformations: Methods of Construction

Preparata [Preparata77] outlines methods for computing the MAT of simple convex and non-convex polygons, in *O(nlogn)* and $O(n^2)$ time, respectively, where *n* is the number of sides of the polygon. An additional *O(n)* operations can find the largest included circle of a polygon. The algorithms use edge removal to recursively reduce the polygon into fewer and fewer sided polygons until only a triangle remains, for which the medial axis is trivial. The algorithm then constructs the full medial axis transformation by combining the binary tree data structure which holds the edges of the medial axes of these recursively generated triangles.

Thinning algorithms are commonly used to generate MATs of images which are discretized into pixels or voxels. The image is stripped of points on its boundary which do not meet specific criteria [Rolland92]. This process uncovers new boundary points, which are iteratively removed until no points are left that are not part of the medial axis/surface. Rolland et al. [Rolland92] use this procedure to find the medial axis for 3D solids; Hjálmarrson et al. [Hjálmarrson94] use a thinning algorithm to determine the moldability of an injection-molded part design.

Samet [Samet83] uses a quadtree data structure and a checkerboard distance transform in order to successively subdivide the image into successively smaller subregions until those subregions have a single gray scale value. By subdividing the image in this manner, Samet's algorithm not only computes the MAT of a 2D gray scale image efficiently, it also provides a well organized data structure for retrieving the original image. This method is a particular example of finding local distance maxima based on a discrete distance mapping of the image, which is, according to Rolland et al. [Rolland92] along with thinning one of the two most significant strategies for obtaining a discrete medial axis transformation. The definition of the medial axis may be generalized beyond the Euclidean distance metric to include any convenient distance metric, such as checkerboard distance which Samet employs.

Lee [Lee82] computes the medial axis for a simple polygon in *O(nlogn)* time by recursively splitting its edge list into successively smaller lists, computing the Voronoi diagram of the edge lists at the bottom level of recursion, and then constructing the full Voronoi diagram of the polygon by combining the partial Voronoi diagrams recursively. Lee notes that the MAT of a polygonized image is a subset of the Voronoi diagram, in which the Voronoi diagram edges that share vertices with exterior edges are removed. This concept extends to 3D faceted solid models: the medial surface is a subset of the 3D Voronoi diagram in which the faces of Voronoi polyhedrons which share an edge with the exterior facets are removed. The medial surface of a convex faceted solid model, which has no exterior facets by definition, is therefore equivalent to its Voronoi diagram.

Ogniewicz and Ilg [Ogniewicz92] also use a Voronoi diagram in order to construct a medial axis; they compute a robust medial axis, which they call a Voronoi Medial Axis (VMA) by computing a Voronoi diagram of the boundary points of a polygon. Once this is done, a regularization of the diagram, based on one of three residual functions, reduces its sensitivity to perturbations among the boundary points.

O'Rourke and Badler [O'Rourke79] use a method similar to a medial axis transform for fitting spheres to various surface points inside a volume. This $O(n^3)$ method checks to see that no points on the image surface are located internal to the sphere being generated, and shrinks the sphere accordingly should this condition prove false. These spheres form a set of largest included spheres for the model. The medial axis of the 3D model, which is the set of centers of all largest included spheres, can be approximated as a polyline connecting the centers of O'Rourke and Badler's spheres.

As Samet [Samet83] shows, other distance metrics, such as checkerboard or city block distances, can be used to determine the points of locally maximum distance to the edge of an image. In addition to using non-Euclidian distance in medial axis transforms, the distances can be weighted in order to provide a more generalized format of the MAT when convenient. Peleg and Rosenfeld [Peleg81] list four such generalizations. First, the SPAN technique approximates the MAT using maximal homogeneous disks which possess a unique center, radius and average gray scale value. Second, the GRAYMAT produces a medial axis using a gray weighted distance formula: the shortest distance between two points is the lowest gray weighted length of any path between them. Third, the GRADMAT produces a score based on the gradient magnitudes between all pairs of points with a point *P* at their midpoint, and assigns the score to point *P*. Finally, Peleg and Rosenfeld introduce the MMMAT, which uses iterated local min and max operations on a binary image. The method is similar to iteratively shrinking and expanding a binary digital image based on an appropriate neighborhood for the distance metric used. These generalized transforms apply to pattern recognition of gray scale images: the weighting of the MAT allows information about the shading the image to be stored within the MAT's data structure, along with information about the image's external geometry.

The analytical medial axis transformation of a 3D object proves to be difficult to calculate and implement. Sudhalkar et al. [Sudhalkar93] compute a skeleton of polyhedron which they claim shares the desirable properties of the MAT. Their skeleton exhibits dimensional reduction, which means that a 3D object forms a skeleton of 2D faces; it has homotopic equivalence, which means that the number of holes in the object is equal to the number of holes in the skeleton; and it is invertible, which means that the shape of the object can be retrieved using the information provided in the skeleton alone. Invertibility implies that each object has one unique skeleton. The skeleton is computed by faceting a solid model, voxelizing the model, and checking each voxel to determine whether it lies on the skeleton of the object. However, since it uses a non-Euclidean distance norm in this determination, it may differ significantly from the actual medial axis [Sherbrooke95].

Sherbrooke et al. [Sherbrooke95] use a recursive algorithm and an entity classification scheme to develop a method of calculating the Voronoi diagram of a 3D polyhedron without the need for voxelizing or otherwise discretizing the volume; it appears to be the only algorithm which does not require discretization of the solid model. The algorithm begins by finding a junction point of the medial axis and traces each medial axis edge, which they call a seam, until another junction point is found. The process then repeats recursively until the medial axis is entirely traced. The tracing scheme uses a differential equation to incrementally advance along each seam.

Sherbrooke, et al.'s algorithm applies to convex and non-convex polyhedrons with simply connected faces; holes are also allowed. This makes the algorithm a significant advancement of the current state of the art in 3D medial axis transformations. The method is in many ways more general than the method proposed in this thesis, and appears more robust. However, coplanar faces are not allowed, which means that this algorithm cannot directly handle standard .STL files as input. Given that coplanar faces are easy to detect and combine, a modified version of this code would provide an excellent way of extending the work of this thesis to treat general, simply connected faceted casting models.

3.2.2 Voronoi Diagrams: Methods of Construction

Incremental construction is a widely used method for producing a Voronoi diagram of a set of points [Sugihara92] [Choset94]. In this method, the Voronoi diagram is constructed for two arbitrary sites selected from among the set of sites. Additional sites are added one at a time, and the Voronoi territories are modified to reflect the inclusion of these new sites. This method can reduce the overall number of operations required for a naive, $O(n^2)$ construction approach, since the insertion of a new site inside an existing territory requires only that the existing territory and its adjacent territories be updated.

Growth algorithms provide another relatively simple approach to Voronoi diagram generation. Morris and Smyrl [Morris89] grow circles about randomly positioned points in order to generate a Voronoi diagram. The circles grow from their respective centers until they are large enough to intersect with another circle. No circle is allowed to impinge upon the area of another circle; hence as the circles continue to grow, they will deform to produce straight line boundaries between contacting circles. These lines become Voronoi polygon edges. This algorithm works well for applications that can be modeled as statistically random sets of point sites, such as galvanic surfaces or heterogeneous materials [Ghosh94] [Ghosh95a] [Ghosh95b] [Morris89].

Voronoi diagrams of point sets are well understood. Many efficient, robust algorithms have been developed for them. Sugihara and Iri [Sugihara92] use the incremental approach to generate a robust Voronoi diagram in single precision arithmetic by ensuring that the calculated Voronoi diagram shares the same topological structure as the true Voronoi diagram. Simply put, the exact equations of intersections of the bisecting lines which form the Voronoi edges are not of concern; rather, the edges generated are constrained to intersect the same edges that the exact edges are expected to intersect. With this topological approach, Sugihara and Iri were able to successfully produce a Voronoi diagram for 2D sets with one million generator points.

Algorithms also exist which compute generalized Voronoi diagrams. Due to the additional issues involved in extending the diagram to treat generalized sites and non-planar spaces, however, this work is less extensive than research on classic Voronoi diagrams. However, a few algorithms have been implemented. For instance, Sugihara [Sugihara93] approximates generalized 2D Voronoi diagrams by representing the sets of objects as a collection of points, generating the ordinary Voronoi diagram for those points, and removing Voronoi edges that border Voronoi polygons between points on the same object.

Stifter [Stifter91] presents an axiomatic approach for providing an exact description of a generalized Voronoi diagram of 3D surfaces. Although the approach applies only to restricted sets of 3D surfaces, it is sufficiently general to handle sets of 3D faceted solid models. Unfortunately, no algorithms have been implemented to compute the Voronoi diagram using this approach.

3.2.3 Applications of Voronoi Diagrams and MATs

Voronoi diagrams and medial axis transformations are both intuitive and abundant in nature. When a child draws his first picture of his family, he will almost always represent his parents, siblings and himself as stick figures. Because these stick figures are simplified medial axis transformations of the human body, it becomes clear that the skeletonization of a figure is a concept so easy to grasp that it serves as one of the first visualization tools we use to understand our world. Evidence of natural processes which create skeletonized structures can be seen in the cracked earth of a parched field [Lam92], or in the ripple interference patterns formed by throwing a handful of pebbles into a pond.

Upon considering a toddler's stick figures, it becomes easy to understand why the first academic interest in MATs and Voronoi diagrams arose in the fields of pattern recognition and machine vision. Due to their tendency to appear in many different fields, however, the interest in these constructs has generated research in such diverse subjects as fractal generation, medical imaging, robot path planning, and oil well modeling. This discussion will survey a variety of applications in these and other fields.

Image representation and pattern recognition are the primary fields of application for MATs and Voronoi diagrams. Researchers in these fields have investigated MATs and Voronoi diagrams for over thirty years. Samet's [Samet83] work on QMATs, Preparata's [Preparata77] work on simple convex and simple non-convex polygon MATs, Rolland et al.'s [Rolland92] work on 3D thinning, and Peleg and Rosenfeld's [Peleg81] work on gray scale MATs seek to use the MAT to store the geometric information about an image or an object in a compressed format.

MATs and Voronoi diagram applications to pattern recognition naturally extend to computer vision systems for autonomous robots and vehicles. Krozel and Andrisani [Krozel90] use a 2D Voronoi graph interspersed with Delaunay regions to represent a mountainous flight path for an autonomous military aircraft. The result is a graph of possible flight paths which always maximally avoid every obstacle. An algorithm can then automatically minimize the path distance from the initial entry point on the graph to the target point.

Autonomous and stationary robot path planning uses Voronoi diagrams to minimize path length while avoiding collisions with workspace obstacles. Choset and Burdick [Choset94] develop a Generalized Voronoi Graph, which is a 1D retract of a 2D bounded space. This retract uses the principle that a Voronoi edge drawn between two obstacles maximizes the relative distance between each of those obstacles in order to generate path planning for a robot in a static environment. Stifter's [Stifter91] 1D retract of a robot's workspace solves the Findpath Problem, which states that, for a sphere inside a bounded space, a collision free path to any point within that space can be found on a Voronoi diagram of that space, provided such a path exists.

In addition to industrial robotics, Voronoi diagrams apply to other manufacturing processes, such as sintering, NC machining, and thermoplastic molding. Tei-Ohkawa et al. [Tei-Ohkawa94] use the Voronoi diagram to study the atomic packing geometries of the crystalline structure of sintered materials. Takata and Tsai [Takata94] use Voronoi diagrams to optimize the tool path of NC machines performing pocket machining operations. Hjálmarrson, et al. [Hjálmarrson94] find the medial axis of an injection mold design in order to generate a well-defined mesh for injection molding finite element analysis. Lee and Lee [Lee95] also apply the medial axis to mesh generation for injection molding analysis. This computation comprises of voxelizing a faceted solid model and checking each voxel individually to see if it includes a portion of the bisecting surface of one or more planes of facets.

The possible applications for Voronoi diagrams in FEA include not only injection molding analysis, but also computational fluid dynamics studies, heterogeneous materials analysis, and, as explained above, evaluation of the manufacturability of cast parts. Taniguchi and Kobayashi [Taniguchi91] develop an unstructured grid based on the Voronoi diagram for analyzing the fluid flow around solid bodies via the finite volume method. The Voronoi diagram provides an easier means of generating a finite volume grid which conforms optimally to the shape of the solid. Ghosh et al. [Ghosh94] [Ghosh95a] [Ghosh95b] developed a 2D finite element called the Voronoi Cell Finite Element which models a heterogeneous material with secondary material inclusions located at the point sites of Voronoi territory shaped elements. This finite element method produces the same accuracy as conventional FEM, but with much fewer elements. Cruz and Patera [Cruz95] also investigate heterogeneous materials using Voronoi diagrams. Based on a Voronoi diagram representation which also models secondary inclusions at point sites, they formulate finite element supercells. A parallel finite element code analyzes these supercells.

In addition to the above fields, which have generated large bodies of research, Voronoi diagrams and MATs have application to an eclectic and extensive array of research subjects. Medical imaging, a specific subset of pattern recognition, employ MATs for reproducing solid objects based on tomographic surface data [O'Rourke79], in analysis of white blood cells and chromosomes, X-ray image analysis, and the study of coronary arteries [Lam92]. Petroleum engineers have used Voronoi diagrams to accurately model the characteristics of multiple-well oil reservoirs [Palagi94]. Materials science researchers have used them to model galvanic cells in order to simulate the galvanic corrosion on a given surface [Morris89] and to model pore size distribution in the sintering process [De Jonghe89]. Electrical engineers have used them, along with Delaunay graphs, to provide an upper and lower bound to the discretization error associated with MOSFET devices [Tanimoto92]. Computer scientists have used them to optimize neural net design [Bose93]. Mathematicians have used them as a vehicle to generate fractal patterns [Shirriff93].

3.2.4 Potential Application of Voronoi Diagrams and MATs to Metal Casting

Although the research into MATs and Voronoi diagrams is extensive, the foundry industry has not yet published work describing the application of these concepts to either solidification modeling or to automated design of rigging systems. This is no doubt due to the fact that no commercially reliable software has been formulated which computes the analytical MAT of general 3D solids. Once this obstacle is overcome, however, a MAT based solidification modeling application should provide several advantages. The model itself should be fast to compute, since it is a non-iterative process. Voronoi diagrams are not exactly equivalent to the MAT for general solids, but would be trivial to compute given the MAT. The Voronoi diagram's application to FEM mesh generators make them attractive to the foundry industry even if the greatest included sphere method itself would prove to rough an approximation for casting applications.

3.3 Automated Rigging Design

The purpose of performing a solidification simulation upon a given casting is to aid in determining the size and location of risers required to make the casting sound. Risers provide a source of make up metal to the casting, which prevents porosity due to contraction during cooling and solidification. However, riser design cannot be performed without also considering the mold parting planes, the gating, and the runner system. Ideally, the casting design process should be entirely automated. With such an automated system, the foundry would accept a solid model of the part to be cast and generate the final rigging system via computer. This section shows, however, that in almost all cases a completely automated system would be impractical to create and use, and would likely prompt formidable resistance from experienced foundry engineers. The reasons for foundry resistance is also included in this section: foundry engineers rely on experience and rules of thumb as much or more than on quantifiable analysis. They use foundry specific design rules, which, because they may address different design considerations, may contradict other rules. After introducing these problems, the following paragraphs will briefly overview casting design expert systems, which comprise a major research focus for automating the design process.

In order for a computerized rigging design application to be successful, it cannot merely be technically correct; it must also overcome the suspicions and traditions found in the foundries that use it. A good example concerns the issue of gating design. Wukovich and Metevelis [Wukovich89] emphasize this difficulty when they point out that gating is "still regarded as an art or religion rather than a science." They further state that, despite fifty years of collected data on gating, it remains a poorly understood foundry process. This poor understanding arises as much from experience-based personal preferences than from a lack of quality scientific data. For example, Jordan et al. [Jordan88] call pressurized gating systems 'suspect' due to their tendency to allow gases to come out of solution while Karsay [Karsay72] contends that pressurized gating systems should be used in most situations to prevent dross carryover. With such contradictions, generating sound, explicit rules for automated rigging design becomes extremely difficult, if not impossible.

While the industry maintains a skeptical eye on automated design systems, some progress has been completed, especially in the use of knowledge-based expert systems. Expert systems are well suited to resolving the difficulties mentioned above, since they accommodate experience-based design criteria, and can tolerate uncertainty in both design rules and data [Durkin94]. Upadhya and Paul [Upadhya93] have developed a FORTRAN language knowledge-based system which determines a parting plane, performs a point modulus solidification simulation, and uses the output of that simulation to position and dimension risers, gates, sprues and runners. Zhang et al. [Zhang94] use an AutoCad geometry-based solidification modeler to automatically design feeders. Hill et al. [Hill91] describe an expert system for sand casting called EXCAST, which, although not specifically designed for automated gating and risering, makes rigging design problems apparent to the user. Kotschi [Kotschi89] provides examples of algorithms required for automated rigging design, particularly parting plane determination, and automatic core design. The above systems use rapid solidification modelers to provide input data to their automated design programs [Pehlke88] [Zhang94] [Hill91]. Hill's input format may also be separately rendered as a solid image which resembles a finite element analysis model of a casting.

3.4 Observations

From the discussion of Voronoi diagrams and medial axis transformations, it is clear that these geometric constructs have a wide range of engineering applications. A major impediment to using these constructs is that the development of robust, efficient algorithms for computing them is non-trivial. However, this field is an extensive, mature research area; therefore, it is reasonable to expect that these algorithms will eventually become commercially available. Once available, these algorithms will have several potential applications in the foundry industry.

A Voronoi diagram or MAT of a casting model can provide two advantages to an automated rigging design expert system developer: 1) they represent a compact, computer-friendly representation of an object's full geometry, and 2) the data required to approximate the solidification pattern is incorporated into the geometric representation.

In effect, the Voronoi diagram provides a means for a computer to interpret the geometry of a solid model in the same way that a human expert would, without the need for the computer to have human-like sensory input capacity. Since expert systems must interpret input data in order to make decisions, they are limited by the ability of the computer to collect that input data [Durkin94].

The practicality of fully automated rigging design systems to the foundry industry remains unclear. Different components of the same rigging design may have contradictory design rules [Hill91], and certainly competing foundries will differ on their choices of design rules. Therefore, and over-ambitious automated design system may be difficult to market. Despite this uncertainty, however, it remains clear that Voronoi diagrams can provide an immediate aid to engineers by helping them to better visualize and analyze the casting solidification process.

The purpose of this thesis is to explore the feasibility of using a Voronoi diagram, without further processing, in order to rapidly approximate solidification patterns in castings. This chapter shows several applications which indicate that rapid solidification approximation is a significant area of interest to foundrymen. The Voronoi diagram also has potential for use in foundry applications which are not primarily concerned with rapid solidification modeling, but rather with accurate finite element analysis. For example, this review shows two applications where Voronoi diagrams were used to generate new finite elements which provide improvements in accuracy and computation time

[Ghosh94][Ghosh95a][Ghosh95b][Cruz95], and one application which uses a Voronoi diagram-based grid for optimizing mesh shape and alignment [Taniguchi91].

Chapter Four

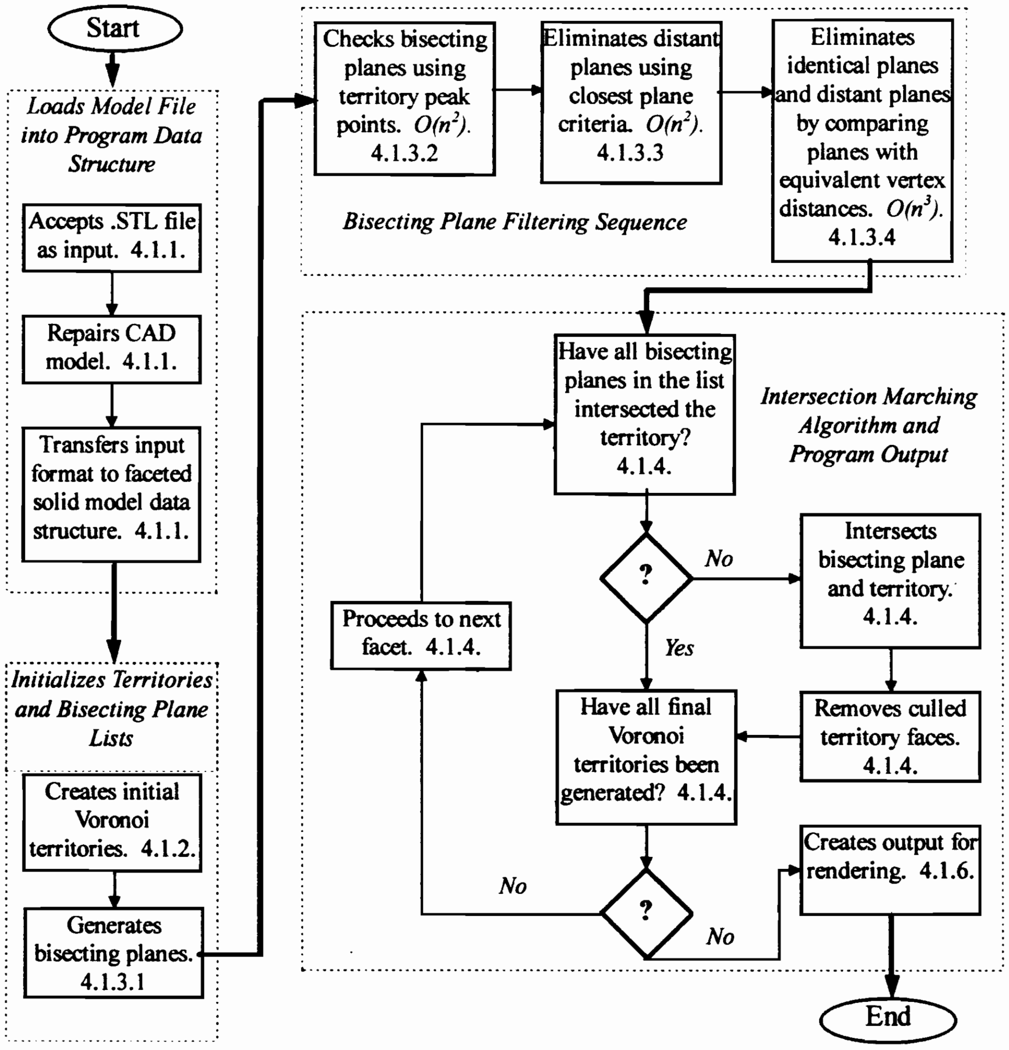
Methods

This chapter considers two main topics: the methods used to generate the Voronoi diagram of convex faceted solid models, and the methods used to investigate the feasibility of applying the Voronoi diagram to casting solidification modeling. The Voronoi diagram generation algorithm described in Section 4.1 combines the intersection marching algorithm described in Chapter Two with a bisecting plane preprocessing sequence in order to generate individual Voronoi territories for each facet in the model. The experiments described in Section 4.2 consider the quality of the generation algorithm, the use of geometry to approximate solidification for basic 2D and 3D geometries, and the algorithmic complexity of the automated greatest included sphere method.

4.1 Voronoi Diagram Generation

This thesis combines a naive *$O(n^2)$* intersection algorithm with an *$O(n^3)$* preprocessing sequence in order to generate a Voronoi diagram of a convex faceted model. Preprocessing is necessary to enhance the robustness of the intersection algorithm. The procedure starts by loading the original .STL model, supplementing it with topological information, and assigning it to an appropriate data structure. It then completes a preprocessing sequence which consists of: 1) constructing initial Voronoi territories of each .STL facet, based on its three adjacent facets; 2) generating, in matrix form, a list of bisecting planes between each facet and every other facet in the model; and 3) eliminating all bisecting planes of each facet's list which can be identified as non-members of the facet's final Voronoi territory. Once the model is preprocessed, a marching algorithm computes the final Voronoi territory of each facet and writes the information required to render each territory to an output file. A separate graphical user interface program renders the .STL model and its Voronoi diagram. This section discusses each of the high-level steps of this procedure in greater detail in order to understand the concepts and methods involved in their completion.

This section will include a C-like pseudocode in order to provide a clearer understanding of the algorithms used. Predefined constants are represented in capital letters, variables are represented in italics, program commands are represented in bold, function names are represented in bold italics, and conditional statements are represented as natural language phrases in plain text. The pseudocode borrows the C++ "//" symbol to denote a single line comment. Figure 4.2 provides the definitions of important variables and constants used in the pseudocode. Others will be introduced as needed.



Start
Loads Model File into Program Data Structure
Accepts .STL file as input. 4.1.1.
Repairs CAD model. 4.1.1.
Transfers input format to faceted solid model data structure. 4.1.1.
Initializes Territories and Bisecting Plane Lists
Creates initial Voronoi territories. 4.1.2.
Generates bisecting planes. 4.1.3.1
Checks bisecting planes using territory peak points. O(n²). 4.1.3.2
Eliminates distant planes using closest plane criteria. O(n²). 4.1.3.3
Eliminates identical planes and distant planes by comparing planes with equivalent vertex distances. O(n³). 4.1.3.4
Bisecting Plane Filtering Sequence
Have all bisecting planes in the list intersected the territory? 4.1.4.
Intersection Marching Algorithm and Program Output
?
No
Yes
Intersects bisecting plane and territory. 4.1.4.
Proceeds to next facet. 4.1.4.
Have all final Voronoi territories been generated? 4.1.4.
Removes culled territory faces. 4.1.4.
Creates output for rendering. 4.1.6.
?
No
No
End

Figure 4.1: High level flowchart of Voronoi diagram generation.

```
constant NUM_FACS          (n)     // Number of facets in the solid model.
constant NUM_VERTS          3      // Number of vertices per facet.
facet    facet[NUM_FACS]           // Array of model's facets.
facet    bs[NUM_FACS][NUM_FACS]    // Bisecting plane lists for each facet.
                                   // bs[l][m] holds the bisecting plane
                                   // between facet l and m, whose normal is
                                   // properly oriented for intersection with
                                   // facet l's territory.
territory terr[NUM_FACS]           // Array holding each facet's Voronoi
                                   // territory.
```

Figure 4.2: Variables and declared constants used for pseudocode subroutines.

4.1.1 .STL Model Loading and Supplementation

The solid model described in the .STL file format is built using any CAD system with both solid modeling and rapid prototyping output capabilities. The application then accepts the model data using a function developed by Bøhn [Bøhn93] which applies topology to the model and repairs it in the event that it is not properly closed. The program transfers information from this input structure to the .STL model data structure discussed in Chapter Two.

4.1.2 Construction of Initial Voronoi Territories

The program constructs initial Voronoi territories of each facet for two reasons: 1) to provide a closed, connected polygon which can be incrementally intersected by planes using the marching algorithm discussed in Chapter Two, and 2) to minimize the size of the solution space for each territory immediately, thus minimizing the number of intersections required to generate the completed territory. The initial territory of a given facet is computed by using only that facet and its three adjacent facets as sites.

As discussed in Chapter Two, the Voronoi territory formed by intersecting the three bisecting planes of a given facet and each of its adjacent facets may take on one of two forms. Whenever all four facets are coplanar, the territory has a prism shape; otherwise, the territory has a pyramid shape. A prism is formed by extruding the facet through an arbitrary distance. Pyramids are constructed by binding the common intersection point of the three bisecting planes and creating the territory faces and edges from that point and each of the facet vertices. Prisms and pyramids are essentially the same structure: a prism is a territory whose apex is an infinite distance from its base.

4.1.3 Bisecting Plane Matrix Generation and Processing

The algorithm incrementally updates each facet's initial territory by intersecting it with the bisecting planes created between that facet and all other facets in the model. If computers were able to complete these intersections in infinite precision arithmetic, each bisecting plane could be naively intersected with its corresponding facet territories, regardless of their orientation. However, floating point errors can cause instability in the intersection of planes and polyhedrons [Figure 4.3]. This instability may cause the algorithm to fail [Sugihara94]. Therefore, each facet's bisecting plane list is filtered relative to the projected distances from each facet vertex to each plane. For most geometries, this vertex distance filtering process removes almost all of the bisecting planes of a territory which will not define a portion of the final territory's boundary prior to performing any intersection operations.

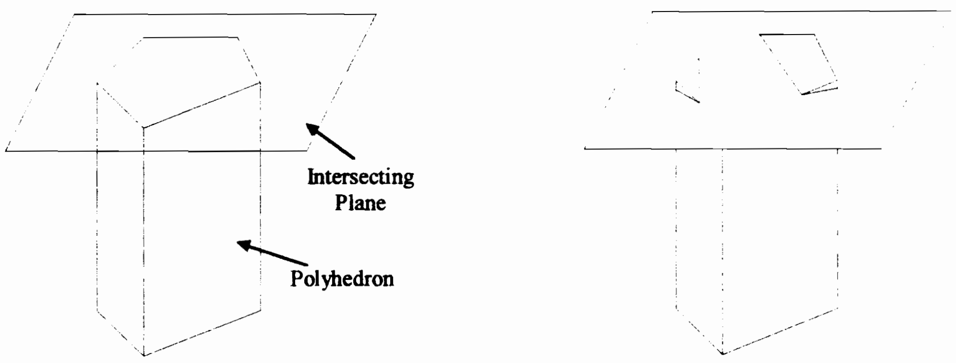


Figure 4.3: Floating point error leads to instability in plane/polyhedron intersections [Sugihara94]. An intersection which should produce one plane cut of a polyhedron may produce two, more than two, or no plane cuts whenever the cutting plane is nearly coincident with a face of the polyhedron.

The following subsections describe each step of the bisecting plane generation and filtering procedure in detail. This procedures starts by creating a bisecting plane list for each facet of the solid model. Then, it check the peak points of each facet's initial Voronoi territory with the bisecting planes in its list and removes those bisecting planes which do not intersect the initial territory. Then, the procedure calculates the distance from each cutting plane to each vertex in the facet, as projected along the facet's normal vector. These distances, called vertex distances for convenience, are then used as a criteria to identify and discard bisecting planes which will not be a part of the final Voronoi territory. The filtering procedure is $O(n^3)$ in complexity, since the elimination of bisecting planes which have one or more equivalent vertex distances with a closer or equivalent bisecting plane is an $O(n^3)$ step. However, this is the third step of the procedure, and the other two steps are $O(n^2)$ [Figure 4.1]. Therefore, in the average case, many of the planes are removed during the two $O(n^2)$ steps, which tend to reduce the size of n during the $O(n^3)$ step.

4.1.3.1 Bisecting Plane Generation

The filtering procedure begins by generating all of the definable cutting planes associated with each facet, and then progresses through a series of successively restrictive checks based upon the bisecting plane vertex distances. No bisecting planes are created between pairs of adjacent facets, and no bisecting planes are created between coplanar facets. The first condition eliminates redundancy, since the bisecting planes of adjacent facets are created and incorporated into their corresponding facet territories during initial territory construction. The second condition is necessary, since no defined bisecting plane exists between coplanar facets. The pseudocode for this section of the program is shown in Figure 4.4.

```
// Double loop:  each facet is paired with each subsequent facet in the model.
for l = 0 to NUM_FACS-1 {
 for m = l+1 to NUM_FACS {

  if (facet[l] and facet[m] are neither adjacent nor coplanar) {
   bs[l][m] = make_bisecting_plane_between (facet[l],facet[m]);
   bs[m][l] = copy_of (bs[l][m]);
   bs[m][l].normal = reverse_direction_of_normal_for (bs[m][l]);
  }

 }
} // End of double loop.
```

Figure 4.4: Pseudocode for bisecting plane generation. ($O(n^2)$ complexity)

A bisecting plane list is generated for each facet in the solid model using this double looped subroutine. If the pair of facets produce a legitimate bisecting plane, then the three lines inside the **if** statement first generate a bisecting plane between the pair which is properly oriented for intersection with *facet[l]*'s Voronoi territory, and includes it in *facet[l]*'s bisecting plane list. Then, the bisecting plane is copied into *facet[m]*'s list, and its normal direction is reversed to properly orient it for intersection with *facet[m]*'s Voronoi territory. At the completion of this subroutine, each facet has a bisecting plane list which comprises of all the possible faces of the facet's final Voronoi territory. Each bisecting plane list requires filtering to remove planes which will not contain faces of the facet's final Voronoi territory.

4.1.3.2 Bisecting Plane Peak Point Checking

Regular geometries, such as cubes and spheres, have a medial surface that degenerates to a single point: the centroid of the object. The Voronoi diagram of a uniformly faceted sphere in infinite precision arithmetic comprises of a set of pyramids whose common apex point is the center of the sphere. Since these pyramids are formed during initial Voronoi territory construction, no further intersections should be necessary to arrive at the final Voronoi territories. However, unless prevented, the program will naively attempt to perform $O(n^2)$ intersections between the facet bisecting planes and the facet territories, which will all degenerate to a single point.

These unnecessary intersections are prevented by inserting the peak point of each initial territory into the plane equation of each bisecting plane; only those planes which fall between the territory peak point and its corresponding facet are kept; the rest are discarded, since they cannot intersect the territory. Additionally, the peak point is perturbed towards the material side of the bisecting plane by an amount proportional to its distance from the facet [Figure 4.5]. This prevents the incorporation of small area faces into the final territory. Small area faces pose two problems: 1) they do not significantly change the territory's bounded volume, but do add to the complexity of its representation; and 2) due to floating point errors, these planes are more likely to cause errors during the intersecting of subsequent planes with the territory.

```
// Specific variables and constants required:
real peak_distance             // Distance from facet its territory peak.
constant SCALE                 // check_peak fn will use the peak distance,
                               // scaled by this factor, to perturb the peak point.
boolean check_peak_result


// Peak point checking requires a full double loop in order to check each bisecting plane.
for (l = 0 to NUM_FACS) {
 for (m = 0 to NUM_FACS) {

  if (bp[l][m] exists) {
   // Calculates the distance from the peak point to the facet.
   peak_distance = get_dist_from_point_to_plane_by_vector
    (terr[l].peak, facet[l], facet[l].normal);
   // Checks the distance from the bisecting plane to the peak point, after perturbing the
   // peak point by a distance proportional to peak_distance.
   check_peak_result = check_peak (terr[l].peak, bp[l][m], peak_distance*SCALE);
   if (check_peak_result = REMOVE_BP) { remove_bisecting_plane (bp[l][m]) };
  }

 }
} // End of double loop.
```

Figure 4.5: Peak point checking pseudocode. ($O(n^2)$ complexity)

At the completion of this step of bisecting plane processing, all bisecting planes which do not intersect their initial Voronoi territory are discarded. The remaining bisecting planes require further filtering in order to identify those which will not be part of the final Voronoi territory.

4.1.3.3 Closest Plane Definition and Distant Plane Removal

The remainder of the filtering procedure initiates by identifying a closest bisecting plane to each facet. The closest plane is defined as the first plane in a facet's bisecting plane list whose set of vertex distances is not strictly greater than the set of facet distances of any other plane in the list. This means that every other plane in the list will either duplicate the closest plane, intersect the closest plane, or lie above the closest plane with respect to the facet, at every point in the space defined by projecting the facet through the entire list of bisecting planes [Figure 4.6].

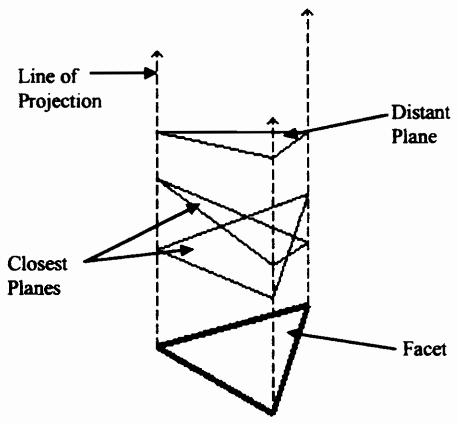


Figure 4.6: Closest planes of a facet's bisecting plane list, based upon projected vertex distances. One closest plane is chosen for each facet list; several may exist.

The closest plane is found by checking each bisecting plane in a facet's list in order; the three vertex distances of each new entry in the list are compared with the corresponding distances of the closest plane found from the previous entries in the list. If all three of the current closest plane's projected vertex distances are greater than those of a subsequent plane in the list, then the subsequent plane becomes the closest plane. This process continues until each plane in the set has been checked [Figure 4.7].

```
// Specific variables and constants required
facet closest_plane[NUM_FACS]  // Creates one closest plane for each
                               // bisecting plane list.
boolean found_one              // When the first closest plane in the
                               // list is found.

// Finding the closest bisecting plane in each list requires a double loop.
for (l = 0 to NUM_FACS) {
 found_one = NO;
 for (m = 0 to NUM_FACS) {

   // Lets the first bisecting plane it finds be the closest plane.
   if (bp[l][m] exists and found_one = NO) {
     closest_plane[l] = bp[l][m];
     found_one = YES;
   }
   // Switches closest plane whenever all of the current bisecting plane's vertex distances
   // indicate that it is closer to the facet than the current closest plane.
   if (found_one = YES and bp[l][m] exists) {
     for (v = 0 to NUM_VERTS) {
       if (the distance from bp[l][m] to facet vertex v >= the distance from
          closest_plane[l] to v) { points_above = points_above +1; }
     }
     if (points_above = 0) { closest_plane[l] = bp[l][m]; }
   }

 }
} // End of double loop.
```

Figure 4.7: Pseudocode for closest plane finding subroutine. ($O(n^2)$ complexity)

Once the closest plane is selected, all distant planes, which are planes whose set of vertex distances are strictly greater than the corresponding vertex distances of the closest plane, are easily identified and eliminated. After distant planes are eliminated, the remaining planes in each list all intersect the closest plane of their list somewhere inside the three projection lines of Figure 4.6.

4.1.3.4 Removal of Planes with Equivalent Vertex Distances

Distant plane elimination, in combination with peak point checking, removes a large portion of bisecting planes in most geometries. All bisecting planes whose projected distance to the facet is strictly greater than that of the closest bisecting plane at every point within the projection lines of Figure 4.6 are removed. However, when one or more of the corresponding vertex distances of a given pair of bisecting planes are equivalent, the possibility exists that one of these planes may also be eliminated from consideration. Three possible scenarios exist [Figure 4.8]:

1. A bisecting plane shares one vertex distance with another, but is otherwise a greater distance from the facet.
2. A bisecting plane shares two vertex distances with another, and thus the two share a common edge somewhere inside the projection lines of Figure 4.8.
3. Two bisecting planes share all three vertex distances with each other. The planes are therefore identical and one can be eliminated.

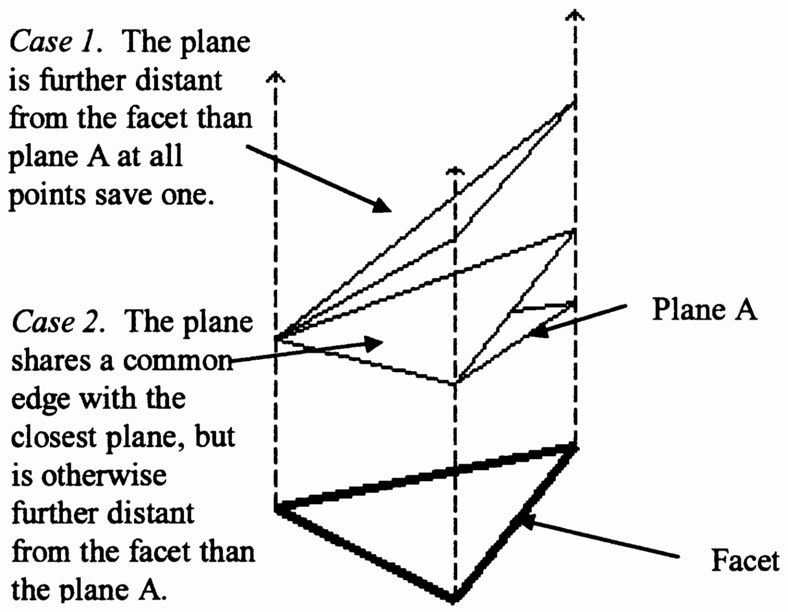

Case 1. The plane is further distant from the facet than plane A at all points save one.
Case 2. The plane shares a common edge with the closest plane, but is otherwise further distant from the facet than the plane A.
Plane A
Facet

Figure 4.8: Planes which share 1-3 vertex distances.

This portion of the filtering sequence is $O(n^3)$, since each of the possible n members of the n bisecting lists are compared with each other in order to identify all pairs of planes which have one or more equivalent vertex distances. However, the first two steps of this sequence eliminate many of the n possible bisecting planes in each list. In other words, if m is the number of planes surviving the first two steps of the filtering sequence, it is reasonable to expect that $m << n$. The pseudocode for this step is shown in Figure 4.9.

```
// This step requires a triple loop.
for ($l$ = 0 to NUM_FACS) {
 for ($m$ = 0 to NUM_FACS - 1) {
  for ($n$ = $m$ +1 to NUM_FACS) {

   if (bp[l][m] and bp[l][n] exist) {
    if (bp[l][m] and bp[l][n] share one equivalent vertex distance) {
     if (bp[l][m]'s other two vertex distances are greater than those of bp[l][n]'s)
      remove_bisecting_plane (bp[l][m]);
     if (bp[l][n]'s other two vertex distances are greater than those of bp[l][m]'s)
      remove_bisecting_plane (bp[l][n]);
    }
   }          // Removes case 1 distant planes.

   if (bp[l][m] and bp[l][n] exist) {
    if (bp[l][m] and bp[l][n] share two equivalent vertex distances) {
     if (bp[l][m]'s other vertex distance is greater than that of bp[l][n]'s)
      remove_bisecting_plane (bp[l][m]);
     if (bp[l][n]'s other two vertex distance is greater than that of bp[l][m]'s)
      remove_bisecting_plane (bp[l][n]);
    }
   }          // Removes case 2 distant planes.

   if (bp[l][m] and bp[l][n] exist) {
    if (bp[l][m] and bp[l][n] share three equivalent vertex distances)
     remove_bisecting_plane (bp[l][n]);
   }          // Removes case 3 (identical) planes.

  }
 }
} // End of triple loop.
```

Figure 4.9: Pseudocode for distant and identical plane removal. ($O(n^3)$ complexity)

At the end of this final step in the bisecting plane processing procedure, all planes which can be removed by considering the projected distances from each bisecting plane to its corresponding facet distances are removed. This processing procedure does not guarantee that all of the bisecting planes which remain in each facet's will contain a face of the facet's final Voronoi territory; this is because the three bisecting planes which form the sides of the initial Voronoi territories are not considered. However, a large majority of the bisecting planes which do not contribute to the final Voronoi territory are removed.

4.1.4 Territory/Bisecting Plane Intersections

A marching algorithm has been designed which intersects each bisecting plane remaining in the matrix with its corresponding initial territory, if in fact an intersection exists. The marching algorithm ensures robust intersections and enforces topology within the territory in most cases. One notable exception includes bisecting planes which intersect the facet at one of its vertices. This exception is resolved by perturbing the point of intersection and redefining the bisecting plane such that it has the same normal, but includes the perturbed point. The marching algorithm is both faster and more reliable than techniques which naively intersect every existing territory edge with a bisecting plane, and reconstruct the territory edge by edge. Such naive methods do not guarantee that the territory will remain closed or that connectivity will be preserved.

After performing any necessary bisecting plane perturbation, the algorithm attempts to find an initial territory edge which intersects the bisecting plane. If none is found, the plane is discarded and the next bisecting plane becomes the current plane. Otherwise, the marching algorithm proceeds as described in Chapter Two and in Figure 2.5.

As the marching algorithm visits a territory face, whether it be to perform an intersection operation or unsuccessfully search for an initial intersecting edge, it records whether that face intersects with, lies above, or lies below the current bisecting plane. Those faces which lie completely above the new face are no longer a part of the territory and are therefore discarded. Those faces which are not visited during the march can only be completely above or below the new face, since any adjacent faces are visited and trimmed during the march. The program inserts one vertex of each unvisited face into the new face's plane equation to determine whether it is above or below the new face, and discards the unvisited face if it is above the new face.

4.1.5 Exception and Error Handling

In order to improve the robustness of the program, the initial territory construction procedure and the marching algorithm have been supplemented with error checking code which detects and corrects three known problems. During initial territory construction, adjacent facets which are almost coplanar, but do not meet the requirements for producing a prism type territory, may produce an apex point which falls on the wrong side of the current facet. A main cause of this problem is that faceted solid models of analytically convex solids may not, in fact, be truly convex themselves. This problem is detected and solved by inserting a newly calculated apex into the base facet's halfspace equation. If the apex is on the incorrect side of the facet, it is reflected onto the facet's correct side, using the facet itself as the plane of reflection. The resulting initial territory acceptably approximates the expected shape of the exact initial territory.

The marching algorithm may unsuccessfully terminate due to one of two situations: once a first intersecting edge is found for a face, a next edge may not be found; or, the initial edge may never be found, and the program will loop endlessly in search of it. Both situations arise when a face intersects the bisecting plane very near to one of its vertices. In the first situation, no second edge is found at all; in the second situation, a second edge is found, but as the march progresses through adjacent faces, it misses the original face because it passes by the initial edge on the other side of the intersecting vertex. Both situations are fixed by interrupting the march, discarding the face under construction, and progressing to the next bisecting plane. The endless loop problem could be fixed by checking for the first occurrence of a duplicate intersection point in the new face's edge list, and trimming its edge list accordingly. This particular solution was not implemented since the error generally occurs only for intersections yielding small area faces. The omission of these faces does not significantly effect the accuracy of the resulting Voronoi territory.

4.1.6 Rendering

The completed Voronoi diagram is transferred into a text file which serves as input to a C++/OpenGL graphical user interface. The interface renders the .STL model as a blue wireframe image, and renders the Voronoi diagram as a solid 3D composite surface. Each face of the Voronoi diagram corresponds to a Voronoi territory face; the GUI assigns a color to each vertex of each face based upon its relative parametric distance from its corresponding facet. The vertex whose distance is a maximum value away from its corresponding facet is assigned a pure red color, while those vertices which coincide with facet vertices are assigned a pure yellow color. The colors of all other vertices are interpolated between yellow and red, based upon their relative distance to the parent facet.

The user has the option of viewing either the full Voronoi diagram of the model or the 'medial surface' of the model, which is redefined solely for the purpose of visualization as the full Voronoi diagram, minus faces which are adjacent to the original facets. The modified medial surface allows the user to view only those areas most likely to be heat centers within the solid model being rendered. The GUI provides a full range of model transformations, but otherwise provides the minimum capability required for visualization and comparison with a finite difference model of the casting geometry.

4.2 Description of Experiments

The Voronoi diagram, in order to be an appropriate tool for casting solidification modeling and design modification, must meet the following criteria: 1) It must be intuitive to visualize; 2) It must predict casting hot spots with good accuracy and speed, and 3) It must generally describe the overall solidification pattern of the model. Since the greatest included sphere method is itself crude, its Voronoi diagram implementation may in fact be crude. However, it will not miss any relevant features, such as local maxima and minima in the temperature profile, nor will it predict any such features which do not exist in the

FDM model. For example, the Voronoi diagram will not predict two isolated hot spots in a model for which a finite difference method predicts only one hot spot.

To evaluate these criteria, the following experiments were performed:

1. The program was run on several simple, yet potentially problematic .STL formatted geometries to see how well the output matched the expected shape of the Voronoi diagram.
2. A 2D geometry criteria for predicting relative temperature was compared with relative temperature as predicted by FDM for a square cross-section.
3. Voronoi diagram models of four 3D geometries were compared with FDM models of identical geometry in order to determine how well the Voronoi diagram matches the solidification pattern as predicted by FDM.
4. Several timed runs were performed upon three selected models using both the Voronoi diagram generator and the FDM simulator in order to better understand the average complexity of the algorithm and to provide a basis for program time comparison.

The following subsections describe each of these four experiments list in greater detail.

4.2.1 Experiment 1: The Suitability and Correctness of The Voronoi Diagram Generation Algorithm

This first experiment examined the following factors affecting the validity of the Voronoi diagram generator output: the effect of varying input geometry, the effect of varying the

.STL facet quantities, and the effect of changing the approach of the .STL file generation in order to preserve convexity and make more regular facets. Voronoi diagrams of eight convex models were generated for this experiment. The basic shapes for these models were: a $2 \times 2 \times 3$ brick, a 45° wedge created from the brick, a block cut by several plane cuts, a $4 \times 1 \times 3$ plate, a sphere, a cylinder, a half cylinder, and a cylinder cut by several plane cuts. Each of these objects were generated using the .STL file format generator of SDRC I-DEAS Masters Series 2.0 [Figures 4.10-17].

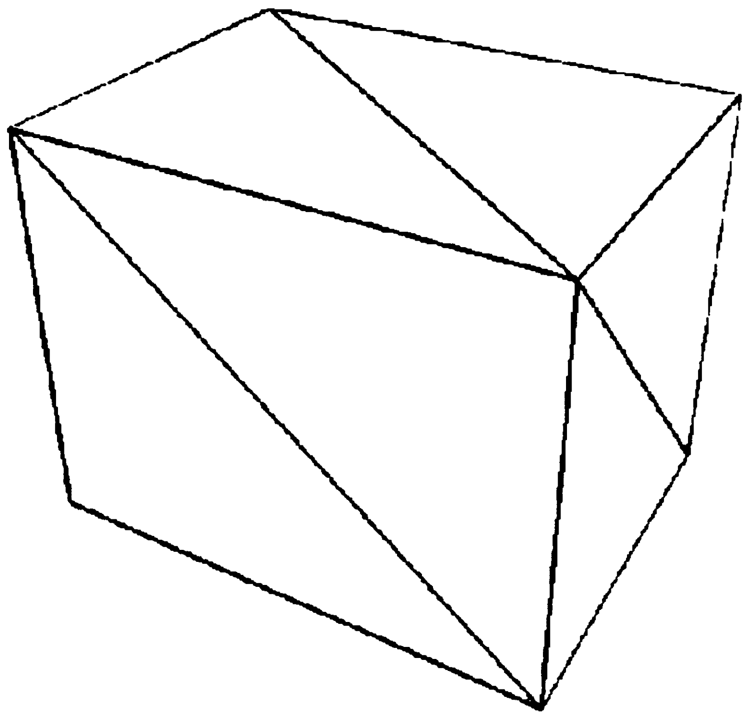


Figure 4.10: 12-facet, $2 \times 2 \times 3$ brick.

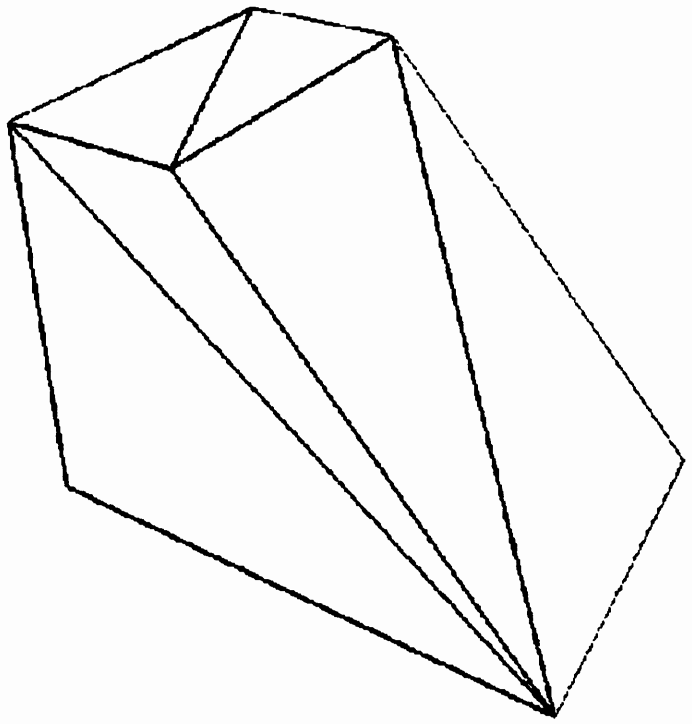


Figure 4.11: 12-facet model of a wedge.

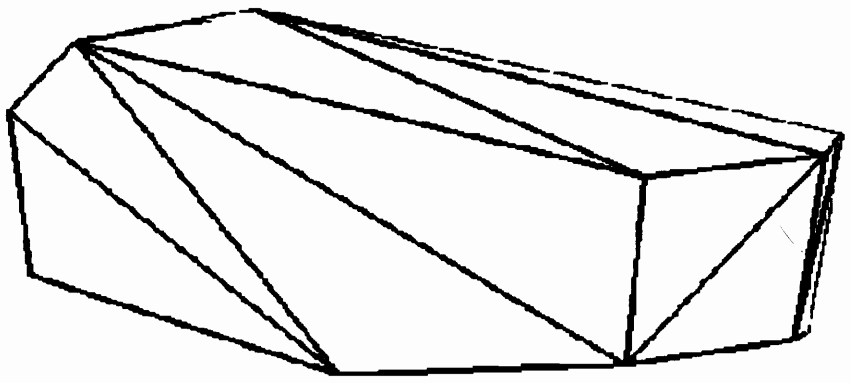


Figure 4.12: 24-facet model of a brick cut by several cutting planes.

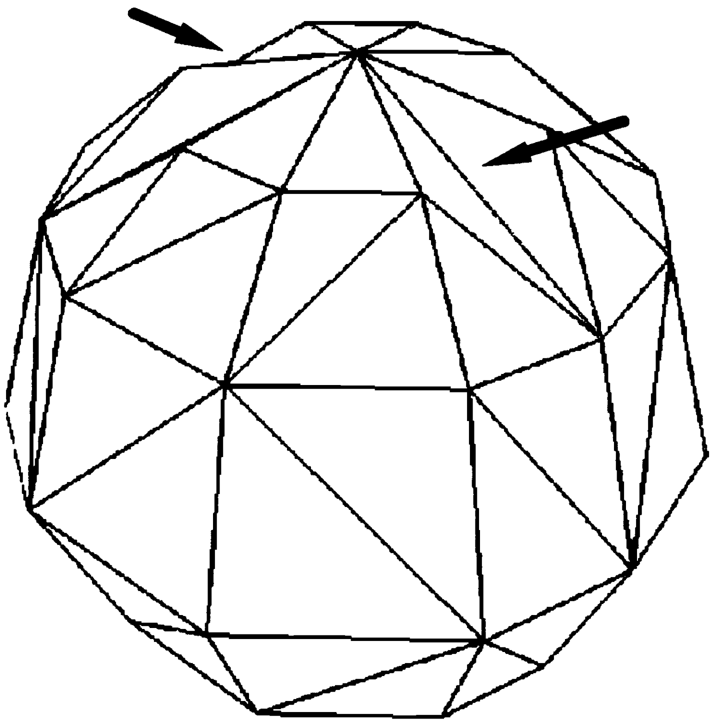


Figure 4.13: 104-facet model of a sphere. Note the non-convex regions.

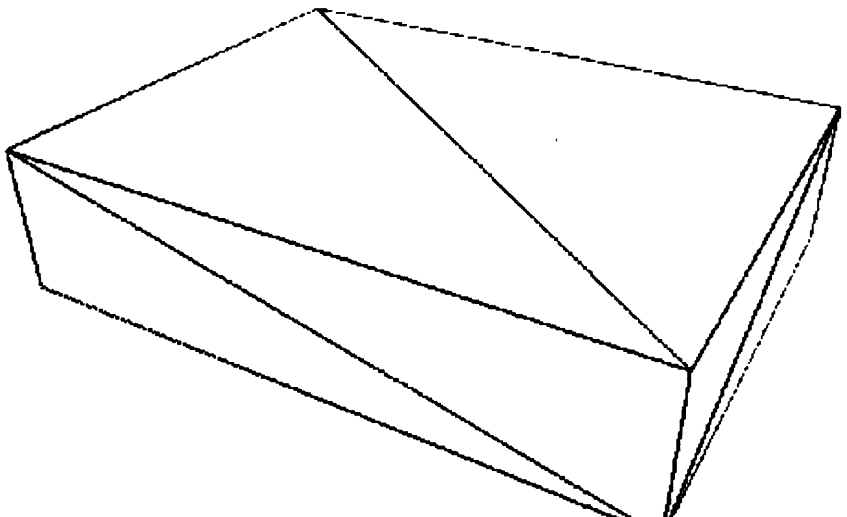


Figure 4.14: 12-facet model of a 4 × 1 × 3 plate.

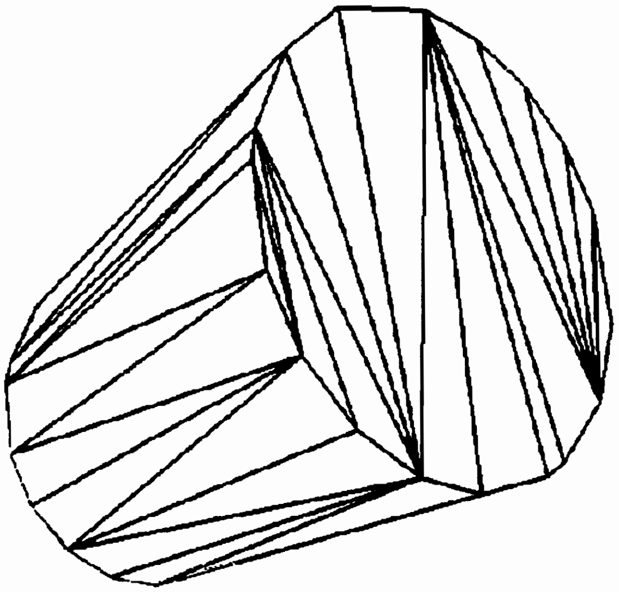


Figure 4.15: 68-facet model of a cylinder.

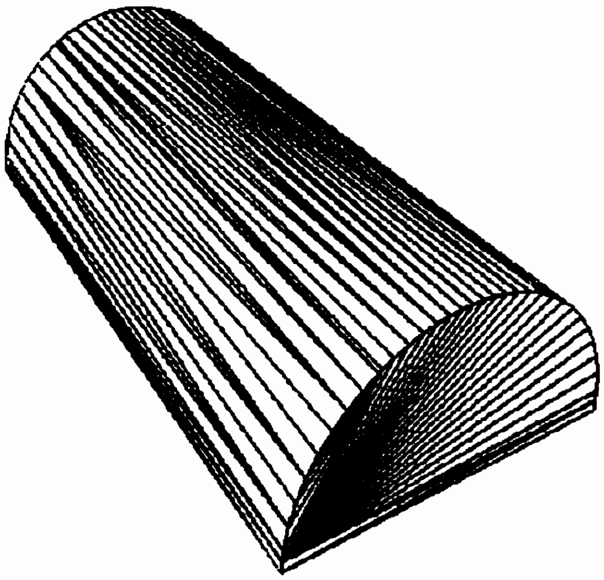


Figure 4.16: 188-facet model of a half cylinder.

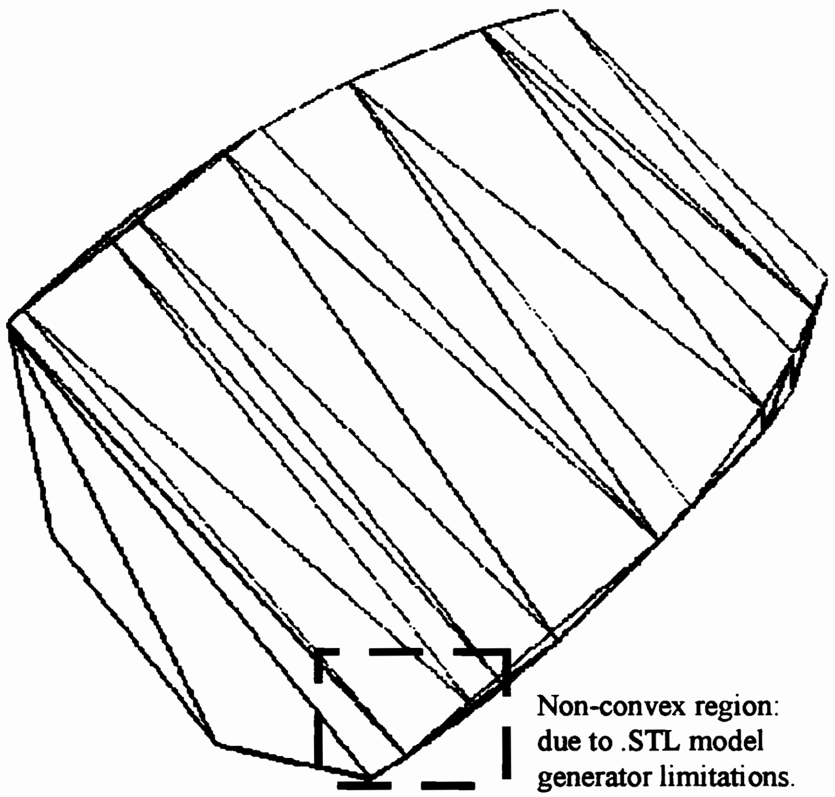


Figure 4.17: 56-facet model of a cut cylinder.

Mathematical verification of the Voronoi diagrams generated from faceted models is non-trivial, since their topologies may differ greatly from those of their analytical model counterparts. For instance, analytical cylinders are convex, symmetric, and have two planar faces and one curved face. Faceted cylinders need not be convex or symmetric, and consist of any number of planar faces which approximate the three faces of the analytical model. Visual inspection was therefore the primary tool used to verify the experimental program's correctnes and suitability for use in solidification modeling, aided by comparisons with the analytical values expected where practical.

4.2.2 Experiment 2: 2D Comparison of Geometry and FDM

The purpose of this comparison was to provide a basic understanding of the capabilities and limitations inherent in using geometry to approximate the temperature of a casting cross-section undergoing cooling. Common heat transfer knowledge [Kreith80] dictates that geometry cannot be used to accurately predict temperature at an arbitrary point within a casting. Should an accurate analysis be required, analytic equations must be used whenever practicable, otherwise, FEM or FDM must be used. However, it is also common knowledge [Kreith80] that symmetry can be exploited to simplify a heat transfer problem, as well as to make basic predictions about its behavior. For example, assuming symmetric boundary conditions, it can be shown that the heat flux normal to a line of symmetry will be zero. Similarly, assuming symmetric boundary conditions, the center of a sphere undergoing cooling can also be shown to be the location of its highest relative temperature. Since the idea of using Voronoi diagrams to model solidification in a casting involves exploiting symmetry in order to provide a more accurate prediction of the relative temperature at selected locations, it is necessary to compare a finite difference analysis of a 2D square cross-section with a general geometry based criteria and to discern which locations within the cross-section show the best correlation between geometry and analysis.

The geometry criterion used to predict the temperature profile is related to the greatest included circle assumption, which states that the relative time needed to cool a given point within a casting corresponds to the radius of the greatest included circle drawn using that point as its center. For a 2D cross-section, this assumption can be expressed mathematically as:

$$\frac{T}{T_{\max}} = \min\left(\frac{r_x}{r_{x_{\max}}}, \frac{r_y}{r_{y_{\max}}}\right), \tag{4.1}$$

where $\frac{T}{T_{\max}}$ represent the temperature relative to the maximum temperature of any location in the cross-section, and $\frac{r}{r_{\max}}$ represents the distance to the boundary of the cross-section relative to the maximum distance from the boundary of the cross section in either the x or y directions. In the case of a square, $r_{x_{\max}} = r_{y_{\max}} = \frac{s}{2}$, where s is the length of a side of the square.

The relative temperature as predicted by this relative distance criteria is computed at each node of a 21×21 square grid FDM model, and compared with the calculated relative temperature of each node. Since the center of the square corresponds to both the location of the highest temperature for a symmetric cooling condition and to the location where $\frac{r}{r_{\max}}$ is a maximum, it is obvious that the difference between relative temperature and relative distance is 0 at the center. Since the Voronoi diagram of a square's four edges is the union of its two diagonals, it is also obvious that the center of a square is on its Voronoi diagram. From this, it follows that the Voronoi diagram successfully shows the hot spot of a square cross-section. The remaining questions become: 1) does the rest of the points Voronoi diagram provide a better agreement between relative distance and relative temperature than at other locations, and 2) is it mathematically valid to approximate the expected cooling pattern of a casting using geometry?

4.2.3 Experiment 3: 3D Comparison of FDM and the Greatest Included Sphere Method

This investigation considered whether the Voronoi diagram geometry can be used to predict the location of hot spots for actual 3D geometries. Unlike the center of the square cross-section above, 3D convex geometries do not always produce a single point hot spot under symmetric cooling conditions; often, the hot spot has a linear or planar geometry. The Voronoi diagram, if successful, should correctly visualize these cases and provide close agreement between the relative distance to the surface of the model and the relative temperature of the model as predicted by FDM in order to be a useful modeling tool. The Voronoi diagram should also show good overall agreement between the relative temperature predicted at its vertices and the relative temperature predicted by the nearest nodes of an FDM model of identical geometry. Additionally, the cooling pattern indicated by the Voronoi diagram must be a mathematically valid approximation of the FDM cooling pattern.

Since the vertices of the Voronoi diagram are the only entities which allow for explicit calculation of relative distance to the surface of the model, the relative distance of each vertex was compared with the relative temperature of its eight nearest nodes in the corresponding FDM model. The value of each individual node is weighted by computing a weighted average of the difference between the relative distance of the Voronoi vertex and each of the eight nodes, as follows:

$$R = \frac{\sum_{i=1}^{8}\left(\frac{r}{r_{\max}} - \frac{T_i}{T_{\max}}\right)\left(\frac{1}{d_i}\right)}{\sum_{i=1}^{8}\left(\frac{1}{d_i}\right)}, \tag{4.2}$$

where $\frac{r}{r_{\max}}$ is the relative distance from the vertex to the nearest surface of the model, $\frac{T_i}{T_{\max}}$ is the relative temperature of node i, which is one of the 8 neighboring nodes of the vertex, and d_i is the distance from node i to the vertex. Using this formula, the smaller the distance between a node and the Voronoi vertex, the larger the weight is applied to the difference in relative temperature between the two. This criteria is more conservative than considering only the difference between the Voronoi vertex and its single closest neighbor, since steep cooling gradients may cause widely differing temperature values between the eight nodes.

4.2.4 Experiment 4: Execution Time Investigations

The Voronoi diagram should ideally provide a fast approximation of the cooling pattern predicted by FDM or FEM, and provide good accuracy in close proximity to the casting's hot spots. This investigation compared the time to calculate the Voronoi diagram of several selected geometries with the time to calculate a temperature profile via the explicit FDM method used in the above investigations. Five grid densities were used to model the brick, wedge and cylinder shown in Experiment 1 [Figures 4.10-12]. The execution time of these models were compared with the time required to generate a Voronoi diagram for models of identical geometry. A second test shows how the algorithm varies with the number of facets by computing the Voronoi diagrams of ten half cylinder models containing 28, 60, 76, 84, 108, 116, 140, 188, 212, and 260 facets, respectively.

As discussed in section 4.1.3, the Voronoi diagram generation algorithm uses a $O(n^3)$ bisecting plane preprocessing routine and an $O(n^2)$ intersection algorithm, where n is the number of triangular facets. This makes the overall complexity of the algorithm $O(n^3)$. Finite element analysis and implicit finite difference methods for solving transient heat transfer problems are also $O(n^3)$, where n is the number of nodes or elements; however, these methods are also iterative. Therefore, a separate $O(n^3)$ operation is required at each time step. The explicit finite difference method is an $O(n)$ method, where n is the number of nodes. However, it too is an iterative method which is not guaranteed to produce stable results. As n becomes large, the time step must become smaller in order to provide an answer that correctly converges, often resulting in a prohibitive number of time steps.

This potential instability also leads to loss of generality, which make explicit finite difference codes impractical for general use.

The questions explored therefore include: 1) do the execution times of the Voronoi diagram models above compare reasonably with the FDM models, or is sparse grid FDM a clearly superior approach to rapid solidification modeling, and 2) how does the Voronoi diagram generator execution time vary with facet quantity on an average basis?

Chapter Five

Results

This chapter considers four major topics in completing the goal of exploring the feasibility of a Voronoi diagram based automated greatest included sphere method for casting solidification modeling. First, the Voronoi diagram generation algorithm is investigated in order to establish the level of confidence in its suitability and correctness. Second, the validity of the assumption that geometry alone may be used to model casting solidification is investigated. This is accomplished by comparing a 2D MAT of a square geometry with a uniform finite difference grid and determining the difference between the temperature distributions predicted by both methods. Then, the question of the validity using geometry to model casting solidification was tested for 3D geometries. Four 3D models were used to compare the relative temperature distributions predicted by the Voronoi diagram based greatest included sphere method and by explicit FDM. The final investigation addresses run time and complexity issues. Several runs of Voronoi diagram models were compared with FDM models of varying grid density in order to provide a basis of comparison of their respective run times. Also, the Voronoi diagram generator computed diagrams for several half cylinder geometries of variable facet quantities in order to determine the average case complexity of the algorithm.

This chapter discusses the results of the four experiments listed above in greater detail. Section 5.1 provides the results of the suitability and correctness checks on the Voronoi diagram generator. Specifically, it tests the validity of the algorithm using basic test geometries, and investigates the algorithm's sensitivity to the number and quality of the facets used to model the input geometries. Section 5.2 shows the results of the 2D comparison between the greatest included sphere method and FDM. Section 5.3 provides these same results for the 3D comparison. Section 5.4 compares the execution time of the Voronoi diagram generator against that of the explicit FDM program. It also provides execution time vs. facet quantity data in order to demonstrate that the Voronoi diagram generator has an average case n^2 behavior, where n is the number of facets.

5.1 Experiment 1: The Suitability and Correctness of The Voronoi Diagram Generation Algorithm

The first step in applying any algorithm to a specific problem is to demonstrate that the algorithm correctly produces Voronoi diagrams of solid models. Experiment 1 evaluates the suitability of the Voronoi diagram generation code used in this thesis for convex faceted solid models by looking at the algorithm's correctness, and by considering its sensitivity to various input geometry issues. Experiment 1 considered the following input geometry issues: varying the overall shape of the model, varying the quantities of facets used to generate the model, and manually improving the faceting automatically provided by the .STL file generator. Since the Voronoi diagram becomes more complex to calculate as the quantity of facets increases, it was numerically verified for a few simple shapes and visually verified for more complex shapes.

This section contains five basic subsections. The first subsection reports upon the validity of the algorithm itself. The second introduces the effects expected from the geometry issues discussed above. The third and fourth report upon the effects of input geometry that can be experimentally changed, namely degenerate Voronoi diagram entities and the quantity of facets in the input model. The final subsection provides concluding remarks on geometry related issues in Voronoi diagram generation.

5.1.1 Validation of the Algorithm

The basic input shapes used to generate the test Voronoi diagrams appear in Chapter Four in Figures 4.10-17. Given that no other available software exists as a standard for computing the Voronoi diagram of convex models, it became apparent that comprehensive verification of the algorithm could not take place: hand calculation is impractical for all but the most simple of geometries, such as the brick and the plate. The Voronoi diagrams of these geometries were hand calculated for comparison with the program. In addition, a 48-facet wedge model was manually constructed in order to compare the normals of its Voronoi faces with those of the identically dimensioned 12-facet wedge.

The plate and brick are both right parallelepipeds, so their Voronoi diagrams are relatively simple to compute. The plate has nine Voronoi faces, one of which lies on its midplane [Figure 5.1]. The brick is a right parallelepiped with equivalent width and height. Therefore, its midplane Voronoi face degenerates into its major medial axis [Figure 5.2].

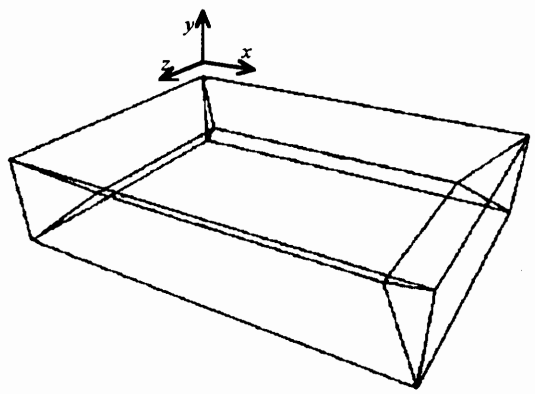


Figure 5.1a: Exact Voronoi diagram of a $4 \times 1 \times 3$ plate.

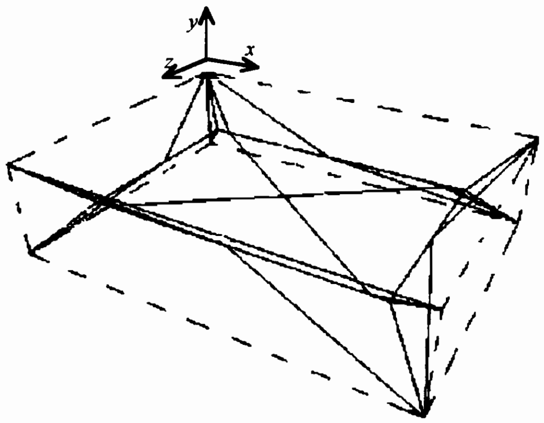


Figure 5.1b: Voronoi diagram of a 12-facet model of the plate. The facets have been removed for clarity.

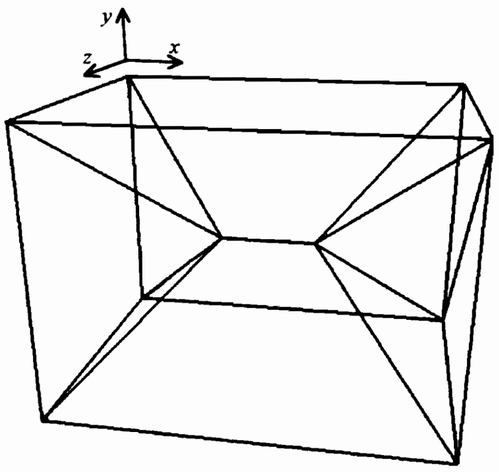


Figure 5.2a: Exact Voronoi diagram of a $3 \times 2 \times 2$ brick.

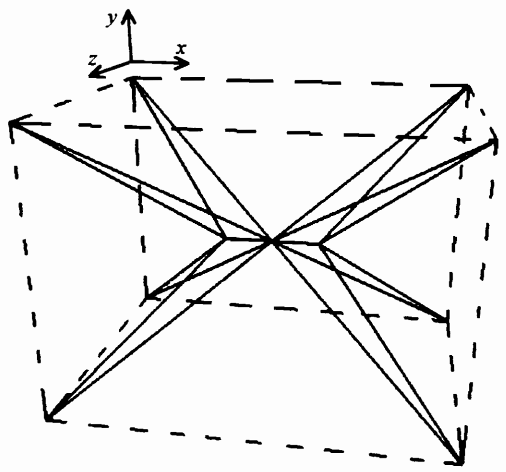


Figure 5.2b: Voronoi diagram of a 12-facet model of the brick. The facets have been removed for clarity.

The program output of Figures 5.1b and 5.2b show additional Voronoi edges than those in the exact Voronoi diagram of Figures 5.1a and 5.2a. These additional lines occur because the exact Voronoi diagrams of the parallelipipeds are constructed using their 6 rectangular faces as sites, while the Voronoi diagrams of the faceted models are constructed using their 12 triangular facets as sites. The 12 facets include more than the minimal information required to model the parallelipipeds. As a result, some of the faces in the exact Voronoi diagrams are represented as multiple coplanar faces in the faceted model.

In order for the Voronoi diagram to be valid, the program must correctly compute the endpoints of each territory edge of the exact Voronoi diagram, along with the normal vectors of each of its territory faces. Additional Voronoi vertices may exist due to faceting the model, provided they lie on a Voronoi face. Additional face normals may not exist. If we ignore as trivial the Voronoi vertices which are coincident with the faceted solid model vertices, then the Voronoi diagram of the 4 x 1 x 3 plate has the expected and program output vertices which are as shown in Table 5.1.

Table 5.1: Voronoi Vertex Data for the 4 × 1 × 3 Plate

Expected Voronoi Vertices			Program Output Voronoi Vertices		
x	**y**	**z**	**x**	**y**	**z**
0.6	0.6	0.6	0.60079	0.60079	0.60079
3.6	0.6	0.6	3.60079	0.60079	0.60079
0.6	0.6	2.6	0.60079	0.60079	2.60150
3.6	0.6	2.6	3.60079	0.60079	2.60079
			Additional Vertices:		
			x	**y**	**z**
			3.43412	0.60079	0.60079
			0.60114	0.60044	1.60185
			3.43507	0.60079	0.60079
			3.60044	0.60114	1.59973
			2.09938	0.60114	2.60044
			2.10220	0.60044	0.60114

Each of the four expected vertices correspond to vertices of the midplane Voronoi face shown in Figure 5.1a. The actual coordinates of the plate are offset by .1 in x, y, and z. Note that no units exist, because the Voronoi diagram is computed without regard to absolute scale. All of the additional vertices coincide with edges of the midplane Voronoi face, and are therefore acceptable. They arise because the plate model has 12 facets for sites, while the exact Voronoi diagram of the plate contains only 6 sites, namely the plate's six faces. The brick's vertex data appears in Table 5.2. The expected vertices are the endpoints of the brick's major medial axis. The additional vertex of the program output is also coincident to its major medial axis.

Table 5.2: Voronoi Vertex Data for the 2 × 2 × 3 Brick

Expected Voronoi Vertices			**Program Output Voronoi Vertices**		
x	y	z	x	y	z
1.1	1.1	1.1	1.10008	1.10079	1.10079
1.1	1.1	2.1	1.10150	1.10079	2.10079
			Additional Vertices:		
			x	y	z
			1.10079	1.10079	1.60079

The vertices of both the brick and the plate are off by as much as 0.00150 due to the perturbation. These differences are therefore expected. Comparison between the exact the Voronoi face normal vectors and the corresponding program output normal vectors shows that all expected normals are represented in program output, no unexpected normals are represented, and the agreement between the exact normal values and the program output are within that expected due to floating point error. Therefore, the program computes valid Voronoi diagrams for these two geometries.

As a further validation check, the 12-facet wedge of Figure 4.11 was manually subdivided into the 48-facet wedge shown in Figure 1.3. The two identically dimensioned models should have the same Voronoi face normals, despite the difference in numbers of facets. A comparison of the program output for both models showed that the 12-facet wedge had 16 Voronoi face normals. The 48-facet wedge shared these normals, but had an additional 3 normals in its output. The additional three normals were due to the 48-facet wedge's coplanar facets; prism type Voronoi faces exist for the 48-facet wedge, but not for the 12-facet wedge. Therefore, the Voronoi diagram of both models remain consistent with each other.

5.1.2 Effect of Input Geometry Upon Program Output

The quality of the Voronoi diagram generated for a given triangular faceted solid model may suffer for several reasons. Voronoi diagrams provide a generalized map of symmetry of an object. Surfaces of symmetry often degenerate into edges of symmetry; similarly, edges of symmetry often degenerate into points of symmetry. When these exceptions arise, any number of territories may share the same edge or point as a boundary entity, leading to problematic intersection operations during construction. Also, poorly generated .STL models may lead to difficulties: the symmetry and convexity of the original model may not be preserved. Without faithfully preserving symmetry, the algorithm will not be able to accurately model the degenerate cases described above. For example, due to a non-uniform approximation of the curvature of a cylinder, its axis of symmetry, which is part of an exact cylinder's Voronoi diagram, may be approximated by a grouping of several Voronoi edges when the cylinder is faceted. Without preserving convexity, the underlying assumption of a convex model is violated, which will produce localized inaccuracies in the Voronoi diagram at the very least.

The next two subsections investigate degenerate Voronoi entities and varying the quantity of facets in the input solid model. It is necessary to approach the geometry issues discussed above by considering these two topics, since the automated .STL data generation greatly affects quality of the results.

5.1.3 Degenerate Voronoi Entities

The experimental Voronoi diagram generator has a limited ability to handle degenerate geometries, such as the $3 \times 2 \times 2$ brick [Figure 5.2] and the 48-facet wedge [Figure 1.3]. Both have degenerate geometries due to bisecting planes between pairs of opposite facing facets. These bisecting planes coincide with the major medial axis of the brick, and coincide with several Voronoi vertices of the wedge. Peak point checking properly eliminates these bisecting planes from their facets' respective lists prior to intersection.

Despite success with these six-face polyhedrons, the program failed to produce accurate Voronoi diagrams of the cylindrical and spherical models. The Voronoi diagram of an exact sphere is simply its center point. The expected Voronoi diagram of an exact cylinder looks essentially like that of the brick [Figure 5.2a]: replace the two four-faced pyramids connected at their peaks in the brick's Voronoi diagram with two 45^O cones. The general axisymmetry of the sphere and the singular axisymmetry of the cylinder, however, make these diagrams problematic to compute with floating point arithmetic. Unlike the brick, however, these degenerate geometries cannot be neatly treated by peak point checking, since the facets of each model do not exactly represent the curvature of their exact geometries.

Figure 5.3 shows the program output of two cylinders. For clarity, the Voronoi diagram faces with edges coincident to the facets of these geometries have been removed from the illustrations. For the 20-facet cylinder, this leaves a V-shaped patch of planes whose common edge closely approximates the axis of symmetry, as is expected. The 68-facet cylinder shows considerable error along its axis of symmetry, however. What should be a single Voronoi edge is a multitude of non-adjacent, parallel edges in the vicinity of the actual axis of symmetry. Some of the cone-shaped portion of the Voronoi diagram is also visible, and appears essentially correct in shape. Thus, these diagrams may still be used by foundry engineers to visualize hot spots; however, they require significant additional refinement in order to be suitable for use as an input for automated design systems.

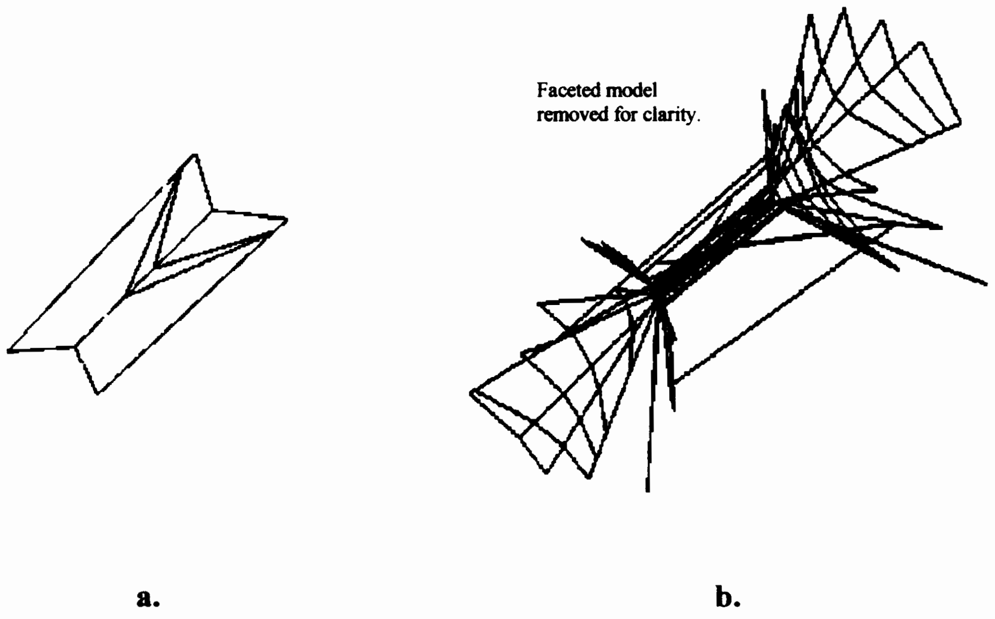


Figure 5.3: Voronoi diagrams for **a)** 20-facet and **b)** 68-facet cylinders. Faces with edges adjacent to facet edges have been removed for clarity.

The poor quality of the cylindrical models arise in part from poor automated .STL solid model construction. While it is true that the .STL files can only approximate the cylindrical surface to within the maximum user-specified absolute facet deviation, the resulting facets lose both the symmetry and the uniformity of curvature present in the CSG cylinder used to generate the .STL files. Indeed, convex, curved surfaces may not produce truly convex faceted models. Since the experimental Voronoi diagram generator assumes a convex input model, this problem may lead to errors during Voronoi territory construction which cause local inaccuracies in the output Voronoi diagram.

To demonstrate the difficulties inherent in poorly faceted .STL input models, a 36-sided extruded polygon was used to approximate a cylinder of the same radius and length as the CSG cylinders fed into the .STL file generator to produce the cylinders of Figure 5.3. Creating this polyhedral approximation of the cylinder prior to input into the .STL file generator essentially tricks the it into producing optimal facets, since it is capable of exactly representing the 36 planar faces which approximate wall of the cylinder with 72 facets. The resultant 140-facet .STL file produced a much higher quality Voronoi diagram than any of the cylinders above [Figure 5.4]. Further refinement of .STL models based upon the exact CSG cylinder caused program crashes for facet quantities above 68.

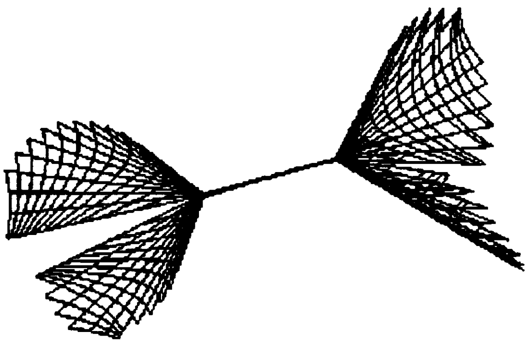
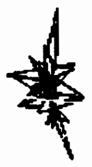
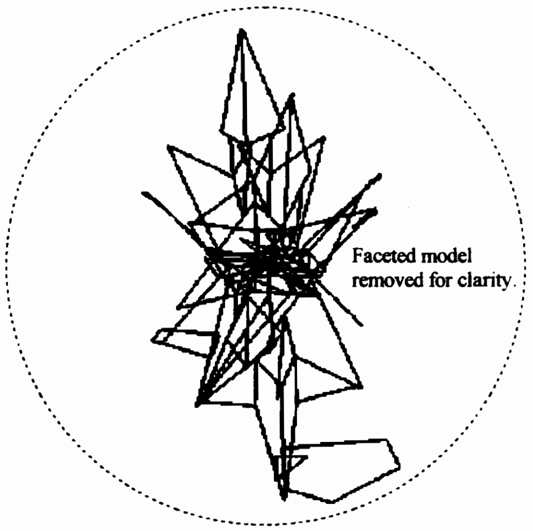


Figure 5.4: Voronoi diagram of a 140-facet polyhedron made to uniformly approximate the curvature of a cylinder. The cylinder facets and Voronoi faces coincident with the facets have been removed for visual clarity. Due to the optimal faceting of this polyhedron, the conic portions of the diagram are much better formed and its major medial axis is almost exactly approximated.

The general symmetry of the spherical models make them even more susceptible to the problems encountered with the cylindrical models. The 48-facet sphere [Figure 5.5a] shows several small planes around its center. This is only a rough approximation of the single point that is expected for an exact Voronoi diagram of an exact sphere, but it provides a reasonable approximation for visualization purposes. The 104-facet sphere, however, is a loose grouping of planes roughly centered about its meridional axis and does not facilitate visual interpretation [Figure 5.5b]. The problems arising from these models can also be traced to the .STL facet generator. Because of the non-uniformity in these models, the Voronoi diagram created for these models varies significantly from the Voronoi diagram of the exact geometries.


a.



b.

Figure 5.5: Voronoi diagrams for **a)** 48-facet and **b)** 104-facet spheres. Faces with edges adjacent to facet edges have been removed for clarity.

5.1.4 Sensitivity of Program Output to the Number of Facets

The .STL file generator used automatically varies the number of facets based upon a user-defined facet deviation value. For models consisting entirely of planar surfaces, like the plates, bricks, and wedges shown, a fixed number of facets can exactly represent the part surface. Therefore, the .STL file generated for these models will contain a fixed, minimal number of facets, regardless of user input. However, curved convex surfaces may cause a loss of model convexity and symmetry during faceting. As a result, the errors inherent in models containing curved surfaces tend to magnify as the facet quantity increases, causing the quality of the Voronoi diagram output to degrade. However, when the representation of the curved surfaces do improve with additional facets, the resulting Voronoi diagram also improves.

The 104-facet sphere [Figures 5.5b and 4.12] and the 56-facet cut cylinder [Figures 5.6 and 4.16] each have localized non-convex regions which affect the output of their Voronoi diagrams. The sphere's diagram is basically unusable. However, the cut cylinder still provides some information since the Voronoi territories are not topologically related. The independent representation of individual territories makes knitting the Voronoi diagram into a single, topologically consistent entity difficult. However, it isolates the effect of calculational errors to single territories rather than to the full model.

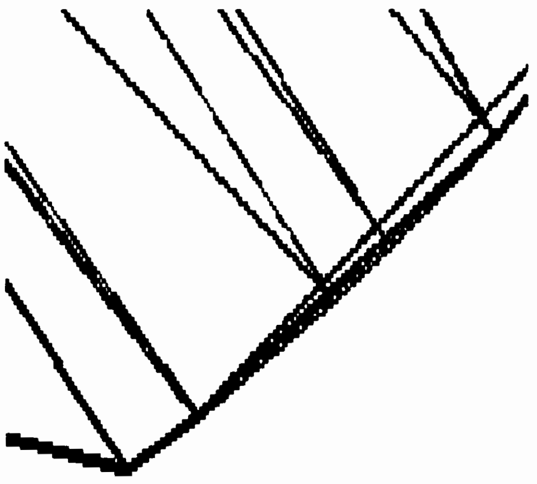
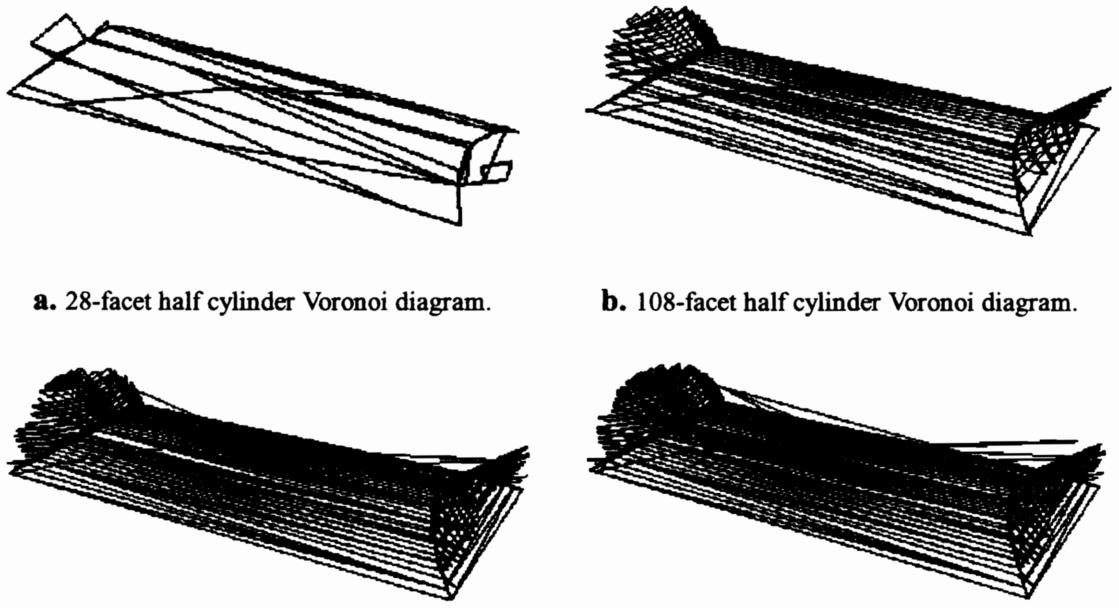


Figure 5.6: Close-up view of non-convex region of the cut cylinder. This shows the limitation of the .STL file generator in modeling geometries with high curvature.

The experimental program generated Voronoi diagrams for 20-, 36-, 52-, and 68-facet cylinders and ten differently faceted half cylinders in order to determine the effects of increasing the facets on output. The cylinders [Figures 5.3a-b] seemed insensitive to facet refinement. The half cylinders, however, showed good initial accuracy and slight, but noticeable improvement with additional facets [Figures 5.7a-d]. The higher quality of the output is due mainly to the fact that the exact Voronoi diagram does not contain degenerate entities, and that the .STL file generator generally preserves the convexity of the original model.

The additional lines which appear in the 188- and 212-facet models do not appear to be correct because they do not lie on one of the five major surface patches of the Voronoi diagram of the half cylinder CSG model which the facets approximate. However, these planes do belong to the faceted models' Voronoi diagrams. Recall that faceted model Voronoi diagrams contain additional information because single analytic surfaces are represented with multiple faces. The trick used to remove the Voronoi faces which do not closely approximate the CSG model's Voronoi diagram, namely to remove all Voronoi faces with a vertex whose point's distance to any facet is lower than some threshold, does not guarantee that the faces left will all lie on the CSG model's Voronoi diagram.



a. 28-facet half cylinder Voronoi diagram. **b.** 108-facet half cylinder Voronoi diagram.

c. 188-facet half cylinder Voronoi diagram. **d.** 212-facet half cylinder Voronoi diagram.

Figure 5.7: Program output for half cylinder models. There exist five major surface patches: the medial surface between the cylindrical surface and the base, the two fan shaped medial surfaces between the cylindrical surface and each of the endcaps, and the two planar medial surfaces between the base and the endcaps.

5.1.5 Concluding Remarks

The experimental Voronoi diagram generator can correctly compute a valid Voronoi diagram for convex .STL model files. However, curved geometries, particularly spheres and cylinders, are problematic for three reasons: 1) their curvatures are poorly approximated by the .STL file generator, and 2) their exact Voronoi diagrams contain degenerate features which complicate a large percentage of the intersection operations used to generate the faceted Voronoi diagrams. True polyhedral models, such as bricks, or even the 38-face polyhedral cylinder of Figure 5.4, are not adversely affected by degenerate Voronoi entities, since their overall number of facets remain low, and those facets exactly model their surface geometries. Degenerate Voronoi diagrams represent the perhaps the most difficult obstacle to overcome in generating a robust, exact Voronoi diagram for general polyhedrons. Currently, degeneracy detection and treatment remains a research issue even for discretized and semi-discretized approaches to 3D Voronoi diagrams and medial axis transforms [Sudhalkar93][Sherbrooke95].

Despite the limitations of the program, it is capable of generating at least visually acceptable Voronoi diagrams of most convex geometries. This makes it acceptable for demonstrating the feasibility of applying the Voronoi diagram to casting solidification modeling.

5.2 Experiment 2: 2D Comparison of Geometry and The Finite Difference Method

The use of geometry to estimate relative temperatures in a model, even a model with uniform boundary conditions and initial temperatures, has obvious limitations. Areas of local heat transfer concentration make using a geometry criterion, such as the greatest included sphere method, inaccurate for general locations inside the model. However, geometry does provide a general understanding of heat flow, and therefore of relative temperature distribution, in heat transfer models. Often this rough understanding is sufficient for initial design iterations for castings.

The FDM model constructed for comparison with a geometry based relative temperature prediction uses a 21 x 21 finite difference grid with a uniform initial temperature of 1200^{O} and a boundary temperature of 800^{O}. These temperatures and their units are arbitrary, since the rules used to relate temperature to geometry are dimensionless. If dimensions applied, then the comparison would lose generality. In addition, the constant in the finite difference equation [Equation 2.9] does not require explicit values. The values used for *C* were arbitrarily selected to be consistent with those of a generic steel. In order to ensure that the complete model was undergoing cooling at the time of program termination, the FDM model was terminated when the maximum relative temperature of the model cooled to 1150^{O}.

Figures 5.8a-c show maps of the relative temperature distribution of the cross-section as calculated by the greatest included circle method, the finite difference method, and the difference in relative temperatures as predicted by both methods. These maps show that the greatest included circle method overpredicts the cooling at every location within the cross-section. The differences in temperature (residuals) are most significant at as the distance from the Voronoi diagram of the model increases. The global maximum magnitude for the residuals is 0.288 at the four points shown in Figure 5.8c. This value corresponds to an underprediction of the FDM temperature by 100.9^{O} on the 350^{O} full range using geometry. The maximum residual magnitude at any point along the diagonals, the union of which forms the Voronoi diagram of the square, is 0.160, which corresponds to a 56.1^{O} underprediction. The center node of the square, which corresponds to the model's hot spot, is correctly identified by the greatest included circle method as the location of maximum temperature; its residual value is 0.000. The average residual magnitude for the whole cross-section is 0.173 (60.5^{o}), while the average magnitude for the Voronoi diagram is 0.085 (30.8^{o}). This shows that, although the Voronoi diagram still provides only a rough estimation of relative temperature, its accuracy is much better than that obtainable for a general point.

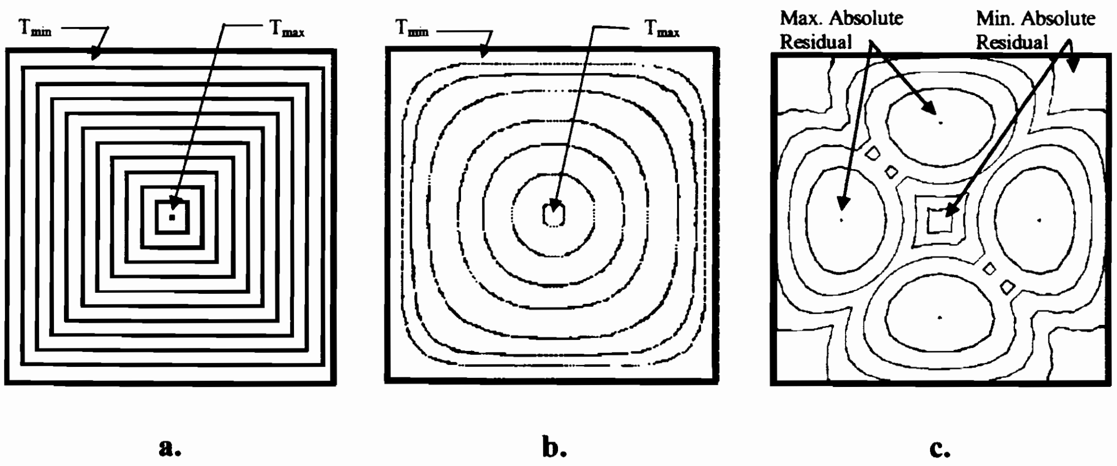


Figure 5.8: **a)** Expected relative temperature profile using a greatest included circle based linear interpolation between center point (T/T_{max} = 1) and edges (T/T_{max} = 0). **b)** Calculated relative temperature profile using explicit FDM. T/T_{max} = 1 at center. **c)** Contour plot of difference between the two methods. Difference is 0 at center and near corners; difference is a maximum at the four peak points located between the center and each of the edges' midpoints.

The reason for the greater accuracy along the Voronoi diagram is that a greatest included circle implies an equal heat transfer contribution from at least two of the cooling surfaces of the model. Ideally, the heat transfer conditions are equivalent at each point on the circle's circumference. The nodes displaying the maximum difference in relative temperature fall on the squares midlines since the heat transfer at those nodes are dominated by their nearest edge to a larger extent than all other nodes in the model. As a result, they cool slower than those nodes near the diagonals, which have at least two significant cooling edges.

The side by side comparison shows several differences between the two models. First, the FDM model contours are pinched at the corners. This indicates additional cooling due to the local increase in heat transfer surface. Second, the variable band thickness of the FDM contours (the thickness increases to a maximum between the third and fourth isotherm lines, then decreases towards the center) and the pattern of the residuals indicate a non-linear relationship between temperature and distance to the nearest edge. However, although the linear approximation ignores the local corner affects and the non-linear behavior of the cooling pattern, it consistently describes the behavior of the FDM temperature distribution: the temperature strictly increases on every linear path from the edge to the center of the square. This shows that while the approximation of temperature by the greatest included circle method is often poor, it can be used locally to determine relative time to cool to a given temperature.

5.3 Experiment 3: 3D Comparison of FDM and the Greatest Included Sphere Method

As the last subsection shows, when the symmetry of a casting geometry is completely understood, the capability of using geometry to approximate a temperature distribution can be greatly enhanced. However, although the overall pattern predicted by this method is consistent with actual cooling pattern as predicted by FDM, its general agreement, even on the Voronoi diagram, is rough. The accuracy provided over the full Voronoi diagram can be expected to provide a reasonable approximation for visualization purposes. However, the most important benefit of using the Voronoi diagram in a 3D casting is to determine where the general symmetry of the object converges to a single point (the sphere of Figure 5.5a), a line (the brick of Figure 5.2), or a plane (the plate of Figure 5.1). This point or set of points correspond to the hot spot of the model, since the distance from the closest casting surface is at a maximum there. Since the location and size of this hot spot is often not obvious, the capability of automatically finding and retrieving it is a valuable asset to casting engineers.

Experiment 3 explored two major issues: 1) the ability of Voronoi diagram to model the cooling pattern of a brick, plate, wedge and cylinder using explicit FDM as a comparison tool, and 2) the accuracy of the Voronoi diagram at the hot spot of the casting.

As described in Chapter Four, the Voronoi vertices of each geometry were compared with the nearest eight nodes of an identically shaped FDM model. Those vertices within a relative distance of 0.1 of the model surface were discarded, since uniform temperature boundary conditions produce unrealistically low calculated temperatures in the immediate proximity of the surface. In order to visually render the results of the comparison, the 8 nearest nodes of each Voronoi vertex were represented as a cube whose color is proportional to the magnitude of its residual. The closer the value gets to zero, the lighter the color of the cube. Negative residuals are represented in green, and positive residuals are represented in red. Since more than one vertex may appear in the same cube, each cube can be viewed individually using the GUI which generated the figures in this section.

5.3.1 The Cylinder

The 36-facet cylinder's extremely rough Voronoi diagram [Figure 5.3] seems to indicate that its usefulness in predicting relative temperature is limited. It is true that the largest magnitude of the relative temperature residual between the diagram and the FDM model is 0.191, which indicates a large disagreement in predicted relative temperatures between the two methods at that location. However, the two models generally provide much better agreement. Of the 67 Voronoi vertices on the interior of the cylinder, 64 had residual magnitudes of .1 or less [Figure 5.9]. Additionally, all nine vertices located on or close to the axis of symmetry of the cylinder show residual values ranging from -0.0160 to 0.0327, with the minimum magnitude residual being 0.000320.

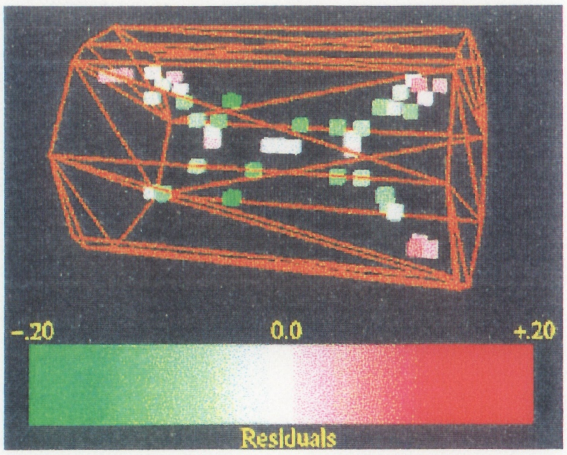


Figure 5.9: Map of Voronoi vertex comparison with 8 nearest FDM nodes. The grid box containing each vertex is shaded with a color representing the residual.

This range of residuals show an important difference between single point hot spots and linear hot spots like those of this cylinder. The Voronoi diagram greatest included sphere method tends to overpredict the relative temperature at the endpoints of the hot spot. This is because, even though the distance from the surface is the same for these points as for other points in the hot spot, a larger percentage of the surface will be closer to the endpoints than to the midpoint of the hot spot. This effect is analogous to the localized effects of internal corners. Therefore, the temperature is expected to be cooler at the endpoints, and hottest at the midpoint, of a linear hot spot. Since the greatest included sphere method predicts that all of the points in the hot spot cool uniformly, it will exactly predict the relative temperature at the midpoint, and overpredict the temperature at all other locations in the hot spot.

The three points which produced the largest negative residuals arise from locally poor approximations of the cylinder's curvature due to poorly formed faceting. As a result, territories which theoretically should have edges or points coincident with the cylinder's hot spot do not come near the hot spot in practice.

5.3.2 The Brick

The brick shares with the cylinder a linear shaped hot spot. Unlike the cylinder, however, the brick can be represented exactly by its 12 .STL facets. This means the Voronoi diagram will be uncluttered and accurate. As a result, its interior vertices should provide a good indication of how well an accurate Voronoi diagram can model cooling patterns in a casting.

Since the brick has only 12 facets, it has much less interior Voronoi vertices than the cylinder: 3 as opposed to 67 [Figures 5.9 and 5.10]. The center point, which corresponds to the centroid of the brick, arises from fortuitous faceting. The two other points, however, will always be present for this geometry regardless of the faceting used. They are the endpoints of the linear hot spot of the brick. The relative temperatures predicted at these two points are therefore more indicative of the overall accuracy of the Voronoi diagram. As expected, each endpoint has exactly the same residual value, 0.0288 between the Voronoi diagram and the FDM model. The positive value of the residual indicates that the Voronoi diagram overpredicts the temperature as compared to FDM, as expected for locations on the hot spot. Since the midpoint happens to coincide with the midpoint of the brick's hot spot, the residual for that point is exactly 0.

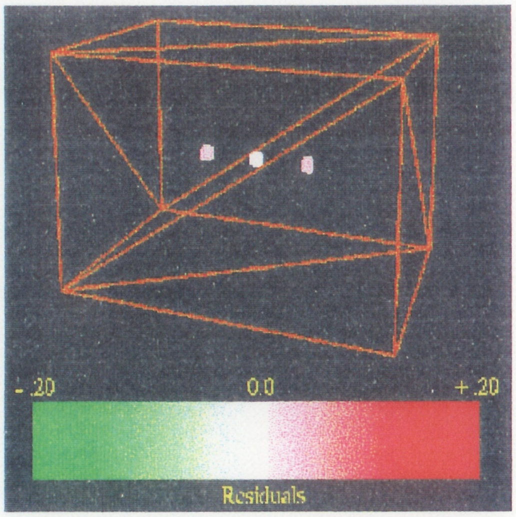


Figure 5.10: Comparison of Voronoi vertex relative temperature values with 8 nearest FDM nodes for the 12-facet brick. The three points fall on the major medial axis of the block, which is also its hot spot.

5.3.3 The Plate

The Voronoi diagram does not predict the exact value of the temperature of a linear hot spot, since the endpoints of the hot spot will cool faster than the midpoint. The same concept applies to planar hot spots; the true location of the hottest temperature will be at the centroid of the plane. Therefore, it is possible that the same end effects which bring down the calculated temperatures of the brick's hot spot endpoints can cause a significant difference in temperature distributed over a large medial plane, such as that of the plate of Figure 5.1. In fact, the expected temperature profile of the square hot spot of the plate should look about the same as that of the square cross-section of Figure 5.8b. However, the difference between the maximum and minimum temperatures of the hot spot should be much less than the difference between the maximum and minimum temperatures of the entire plate. Therefore, the Voronoi diagram should still provide a good approximation of the temperature across the entire surface of the hot spot.

Figure 5.11 shows this assumption to be valid. The plate has 12 Voronoi vertices, all of which lie upon the outline of the planar medial face shown in Figure 5.1a. Of these 12 points, only the four corners are independent of the faceting of the model, and are therefore the only points that should be considered when judging the quality of data provided by the Voronoi diagram vertices. Each corner shows a residual of 0.126, which is high compared to the other models, but still much smaller than the full relative temperature range of 1.0. The large temperature gradients between each of the 8 nearest

FDM nodes is high for each of the corner vertices, which magnifies the difference between the two methods. Once again, the values of each of the corner points exactly match, which shows that the symmetry of the Voronoi diagram corresponds to the symmetry of the FDM temperature distribution.

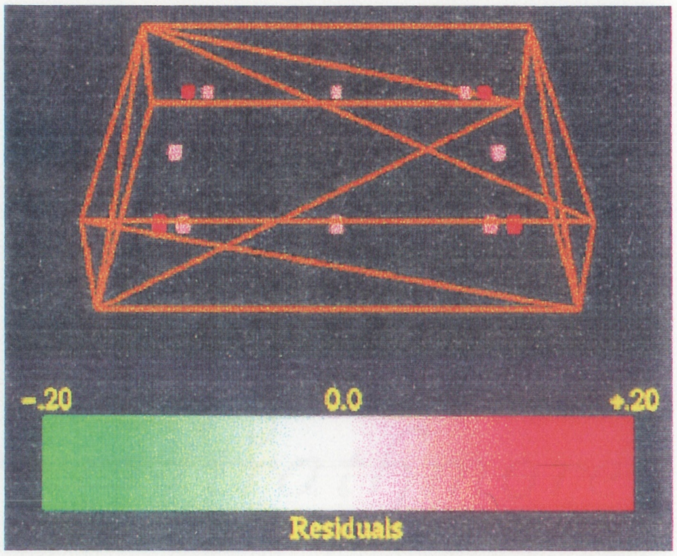
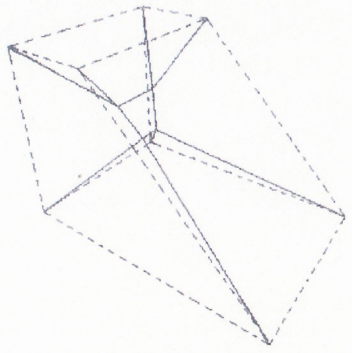


Figure 5.11: Comparison of relative temperature difference between Voronoi vertices and 8 nearest FDM node neighbors for the 12-facet plate. The agreement between these two methods is good despite the temperature profile of the large planar hot spot.

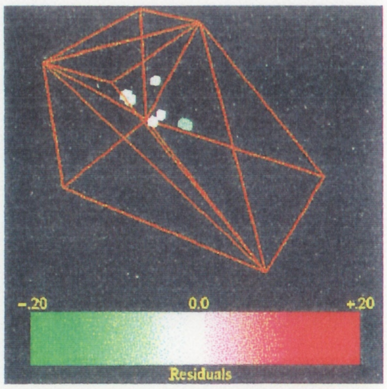
Since the symmetry of the plate forces essentially the same temperature profile between its square hot spot and the square cross-section of Figure 5.8b, the exact location of the maximum temperature can be found by constructing the Voronoi diagram of the planar hot spot. Once the planar hot spot is identified, its full Voronoi diagram can be computed in *O(nlogn)* time [Preparata77]. Similarly, the exact location of maximum temperature in a linear hot spot may be trivially determined by calculating its midpoint.

5.3.4 The Wedge

The wedge model of Figure 5.12 demonstrates how the Voronoi diagram of an object may change radically with a relatively minor change in geometry. Although the wedge is similar to the brick, its Voronoi diagram differs greatly from that of the brick [Figure 5.12a, 5.2]. Its hot spot corresponds to the bottom edge of the small trapezoidal plane in Figure 5.12a; the endpoints of this edge appear as the light pink cubes of Figure 5.12b. The short length of the hot spot relative to the dimensions of the cube means that the temperature variation across the hot spot will be small. Therefore, the endpoints of the hot spot are very close to the overall maximum temperature of the model. The residuals of these endpoints are 0.00948 and 0.00954, respectively, which shows very close agreement between the two methods. Unlike the previous two Voronoi diagrams, these numbers are not in exact agreement because the faceting of the model is not symmetric. The residuals of the other 6 Voronoi vertices were all less than 0.065 in magnitude.



a.



b.

Figure 5.12: **a)** Voronoi diagram for the wedge of Figure 5.11, with its 6 faces used as sites. The faces are dashed to provide visual clarity.
b) Relative temperature comparison between Voronoi vertices and 8 nearest FDM nodes. The two pink boxes contain the vertices which coincide with hot spot endpoints.

5.4 Experiment 4: Execution Time Investigations

As mentioned in Chapter Four, the experimental Voronoi diagram generator is an *$O(n^2)$* algorithm with an *$O(n^3)$* preprocessing procedure, so overall the algorithm is expected to perform faster than the iterative *$O(n^3)$* algorithms used for transient heat transfer finite element analysis and implicit finite difference analysis. Since the explicit FDM method used for this thesis is *$O(kn)$*, where k is the number of time steps and n is the number of nodes, the Voronoi diagram generator should perform much slower than the explicit FDM method as both the number of .STL facets and FDM nodes get large. However, since the approach is intended for initial conceptual design, there should be very little reason for intricately faceted models. The remainder of this section shall consider two major issues: 1) the execution time differences between the Voronoi diagram generator and the explicit FDM program for the wedge, brick and cylinder models, and 2) how the Voronoi diagram generator works on an average case basis. An IBM RS/6000 model 350 workstation with 64 Mb RAM executed the timed program runs used to evaluate these issues.

The following subsection reports the results of timed runs of the brick, cylinder, and wedge models using both the experimental Voronoi diagram generator and the FDM program using dense models. This information is useful to provide an order of magnitude comparison. The complexity of the Voronoi diagram algorithm is compared to that of explicit FDM, implicit FDM, and FEA. The second subsection shows the results of performing timed runs on 10 half cylinder models of variable facet quantities in order to establish the average case complexity of the algorithm.

5.4.1 Execution Time Comparison Between the Explicit Finite Difference Method and The Experimental Voronoi Diagram Generator

This experiment compares the execution times of the Voronoi diagram generator and the explicit FDM code for three basic geometries: the 12-facet brick, the 12-facet wedge, and the 68-facet cylinder. The average execution times for each of these models were 16.6ms, 16.0ms, and 635ms, respectively [Table 5.3]. The brick and the wedge come from 6-sided constructive solid geometry models, and are therefore represented exactly by 12 facets. The 68-facet cylinder was the most densely faceted cylinder successfully run through the code. Therefore, it gives the largest available time to compare with the cylindrical FDM models.

Table 5.3: Execution Times For The Experimental Voronoi Diagram Generator

Model	**Facets**	**Execution Time (s)**
Brick	12	0.0166
Wedge	12	0.0160
Cylinder	68	0.635

Five Explicit FDM models of each of these geometries were also executed and timed [Table 5.4]. The time step size of remained the same throughout the test runs and each run terminated at the same maximum node temperature in order to isolate the effect of changing the node quantities upon geometry.

Table 5.4: Execution Times For The Explicit Finite Difference Method

Model	Nodes	Iterations	Execution Time (s)
Brick	96	74	0.18
	768	271	4.89
	2592	259	15.84
	6144	251	39.02
	12000	247	75.68
Wedge	72	190	0.60
	576	271	6.47
	1944	242	17.13
	4608	213	34.63
	9000	192	56.43
Cylinder	60	147	0.30
	624	233	4.61
	2016	214	13.90
	4992	218	37.28
	9480	209	69.04

Comparing the results shown in Tables 5.3 and 5.4 is non-trivial. The Voronoi diagram based analysis uses a few strategically well placed vertices upon which computations are made. The FDM analysis, on the other hand, relies upon a dense grid of nodes in order to achieve sufficiently accurate results. A direct comparison of these two methods would require a specific maximum accuracy criterion for the FDM analysis as compared to an accepted standard. Only with a specific accuracy criterion in place can an FDM grid be optimized for execution time. Despite this difficulty in comparing execution times, several observations can be made by studying the analysis of the three models.

The first observation is that in general, both methods involve a trade-off between execution time and accuracy in modeling the part geometry. The main difference is that FDM relies upon its the density of its internal nodes to accurately resolve the temperature distribution in the model's interior. The Voronoi diagram, on the other hand, is computed in an exact form from the faceted solid model. Its resolution therefore does not depend upon the number of facets in the model. For instance, if the facets of a given solid model were increased by subdividing each facet in a coarser version of the model, as was the case with the 12-facet and 48-facet wedges of Experiment 1, their Voronoi diagrams would remain consistent. Therefore, once a faceted model is deemed an acceptable approximation of the original casting geometry, no further refinement is needed. An FDM model, on the other hand, may still need extensive additional internal node refinement beyond that required to that accurately approximate the surface geometry of the casting in order to produce an accurate result.

This difference between the two methods is further evident when studying the brick model. As expected for a valid discretized modeling method, the solutions of each of the five FDM models converge towards the exact expected temperature distribution of the model. As a result, the shape and location hot spot predicted by the solutions converges towards that of the hot spot predicted by the Voronoi diagram of the 12-facet brick. Since the surface area of the brick is approximated exactly by its 12 facets, the Voronoi diagram generated those facets is very accurate as well.

A second observation is that the type of FDM nodes chosen by a given application may not be suitable for accurately approximating complex geometries. For example, even the 9480-node FDM model does not represent the shape of an exact cylinder better than the 68-facet cylinder [Figure 5.13]. Because the FDM application used for this thesis is limited to cubic nodes, the curved surface of the cylinder is difficult to approximate with a small number of nodes. As discussed in Chapter Three, FDM is restricted to regular grids of nodes, which makes it less flexible at handling complex geometries than other solidification modeling methods. Therefore, we can often expect that a faceted solid model will require much less facets to accurately represent a casting than the number of nodes in a comparably accurate finite difference grid, with a corresponding savings in execution time.

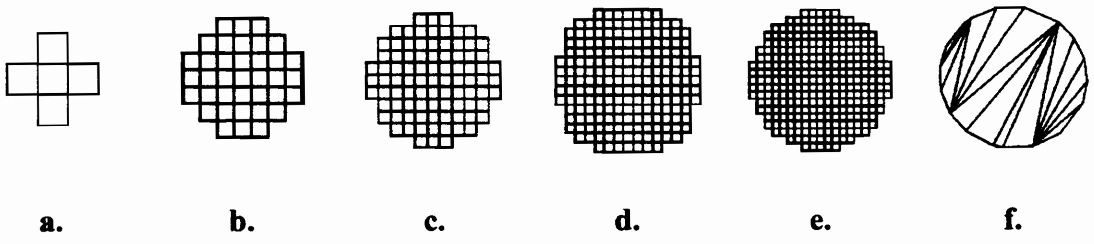


Figure 5.13: 2D cross-sections of 3D FDM grids which approximate the circular cross-section of a cylinder for the **a)** 60-node, **b)** 624-node, **c)** 2016-node, **d)** 4992-node, and **e)** 9480-node models. **f)** The endcap of the 68-node faceted cylinder model compares favorably with the 9480-node model in accuracy.

Finally, the speed of the FDM method also depends upon the size of the time step used. Since it is an iterative method, the speed of the solution may be improved by increasing the time step, at the expense of accuracy. For rapid, approximate solidification, larger time steps may be acceptable; however, whenever the choice of the time step size is left to the user, the possibility exists that the step size selected may lead to an unstable result. In this case, the solution must be rerun at a lower step size. As the model becomes more densely gridded, the step size must be lowered accordingly in order to ensure a stable solution. As a result, the *O(n)* explicit FDM algorithm may become slower than an $O(n^3)$ implicit FDM algorithm due to the need for an excessive amount of iterations to complete a solution. Therefore, as *n* becomes large, the execution time of the non-iterative $O(n^3)$ Voronoi diagram algorithm should still compare reasonably to the explicit FDM method, despite its greater complexity.

In conclusion, the execution times of the two methods do not allow for a specific statement to be made about which has the superior speed. Explicit FDM has the advantage in algorithm complexity. However, modeling issues tend to mitigate this advantage, if not eliminate it. Since the purpose of this thesis is to explore the feasibility of using a Voronoi diagram based model to rapidly approximate solidification modeling, it is not necessary to show that it has superior speed. It is only necessary to show that its speed is not inferior enough to eliminate the method from consideration as a rapid solidification modeler. The results shown in Tables 5.3 and 5.4 indicate that the method has the potential to provide rapid execution times.

5.4.2 Average Case Algorithm Complexity

The Voronoi diagram generator's bisecting plane preprocessing stage has one $O(n^3)$ step, which makes the overall algorithm $O(n^3)$. However, as discussed in Chapter Four, these steps are subsequent to two $O(n^2)$ steps which act to reduce the overall size of n by discarding undesirable bisecting planes. As a result, the overall complexity of the program is expected to be better than $O(n^3)$ for the average case.

The estimated complexity was verified by computing the Voronoi diagram of ten half cylinder models of varying facet quantities [Figure 5.14]. The logarithmic plot of the execution times shows a slope of approximately 2.20, which suggests an average complexity of $O(n^{2.20})$ rather than the theoretical $O(n^3)$. Additionally, the logarithmic plot of the execution times shows a slope of approximately 2, which is expected for a quadratic relationship between execution time and facet quantity.

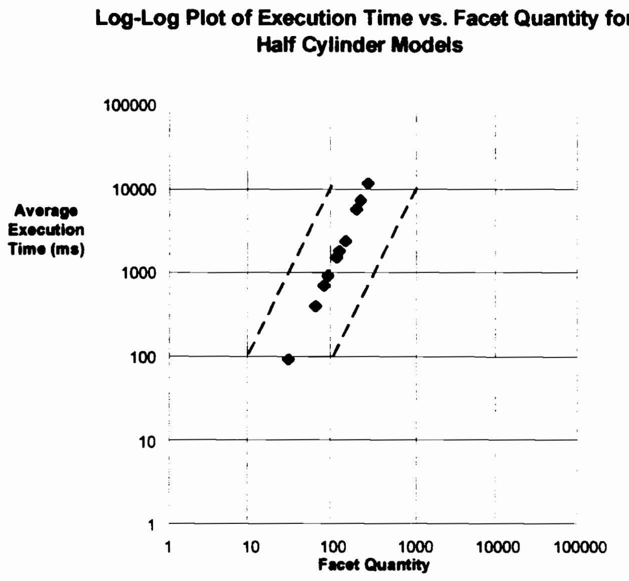


Figure 5.14: Growth rate of Voronoi diagram generator's average execution time due to increase in facet quantity of input model. A curve fit of the data indicates an average complexity of $O(n^{2.20})$. The dotted lines have a linear slope of 2.0.

Chapter Six

Conclusions

This thesis considers the feasibility of using a medial surface transformation to provide a rapid approximation of the solidification patterns of castings. In order to demonstrate feasibility, three issues must be addressed.

1. The medial surface transformation's predicted temperature distribution must be consistent with a well established solidification modeling method. Although an accurate approximation is ideal, consistency is the most important issue when considering the potential for acceptance of this approach.
2. Since speed is an issue, the execution time required to generate the Voronoi diagram must be comparable to an alternative rapid modeling approach. Also, the application must be easy to create input for and use: programs which are fast, but require extensive manual effort in the preprocessing stage, defeat the purpose of using a rapid solidification modeler for multiple design iterations.
3. The medial surface transformation of a general faceted model must be feasible to generate. Since this thesis does not undertake this particular problem, it must be

reasonable to assume that this problem is solvable, and that there is an active interest in solving it.

This thesis shows that the medial surface transformation is, in fact, a useful data structure to apply to casting solidification modeling. Its predicted relative temperature solution remains consistent with explicit FDM solutions for several test geometries. Its speed is equal to or superior to sparse FDM solutions for these same geometries, and the preprocessing issues are easier for the user to resolve for the medial surface transformation code than for the explicit FDM code. Finally, although the Voronoi diagram generator used in this thesis is experimental and only computes Voronoi diagrams for convex faceted solid models, the literature review shows that current research will eventually produce a robust Voronoi diagram for general faceted solid models.

The remainder of this chapter is organized into four sections. The first section discusses the information found in this research which allow the conclusion that the medial surface transformation provides a feasible approach to rapid approximations of solidification patterns in castings. The second subsection discusses the potential advantages to taking such an approach, and the third subsection outlines the obstacles to be overcome. The fourth and final section provides suggestions for further research.

6.1 Conclusions: Feasibility Issues

The medial surface transformation models the solidification pattern in castings with adequate accuracy, and predicts a pattern which is consistent with that of an FDM model.

The 2D comparison of the Voronoi diagram and FDM predicted temperature profiles of a square showed that geometry alone can be used to provide a rough estimation of the relative temperatures in a casting. Additionally, it showed that the Voronoi diagram provides a more accurate prediction: on average, test points on the Voronoi diagram showed slightly over twice as good an agreement with FDM than test points which did not lie on the Voronoi diagram. 3D test models showed that the Voronoi diagram can be used to accurately locate and dimension point, edge, and plane type hot spots in convex geometries. A study of the predicted solidification patterns of FDM models of successively denser meshes converged towards the hot spots predicted by the Voronoi diagram greatest included sphere method, which further adds confidence in the validity of the approach.

The experimental Voronoi diagram generation algorithm also has the potential to rapidly model solidification patterns. Its theoretical $O(n^3)$ complexity is comparable to the iterative $O(n^3)$ complexity of implicit FDM and FEA. Its experimentally measured $O(n^{2.20})$ complexity is indeed better than the complexity of implicit FDM and FEA. Explicit FDM is an iterative $O(kn)$ algorithm, which makes it asymptotically faster than the Voronoi diagram algorithm. However, as n gets large, the time step size of the explicit FDM method required to produce a stable result often becomes small enough to merit using the implicit method.

Timed runs of brick, cylinder and wedge models showed that the Voronoi diagram produced completed models in a comparable time to sparse explicit FDM. Since the use of sparse gridding in finite difference analysis itself constitutes an established rapid solidification modeling approach, the Voronoi diagram provides a reasonable approach to rapid solidification modeling from an execution standpoint.

6.2 Potential Advantages of the Voronoi Diagram Based Greatest Included Spheres Method.

The Voronoi diagram can be produced from geometries represented in a .STL file format, which means that solid model preprocessing can be reduced to a simple file translation inside the CAD system used to create the model. Since time spent preprocessing the model can be as or more important a consideration as algorithm speed when establishing a rapid modeling approach, the advantage of simple preprocessing on any number of commercial CAD systems is an important benefit of this approach. Additionally, a faceted solid model can exactly represent a polyhedral solid model in a fixed number of facets, which means that any purely geometry based model, such as the Voronoi diagram, can be produced without discretization errors for polyhedral models. If the models have curved surfaces, the triangular facets of a faceted solid model provide much more flexibility than the regular volume elements of an FDM grid to model the surfaces accurately. Figure 5.11 shows how this flexibility can lead to accurate models with much lower element quantities: 68 triangular facets provide just as much or more accuracy in modeling a cylindrical geometry than a 9480-element FDM grid.

Just as the geometry of the faceted solid model provides an advantage over FDM grids on the surface of the model, the geometry of the Voronoi diagram provides an advantage over FDM grids on the solid model interior. The Voronoi diagram of a faceted solid model is an exact geometric structure. Therefore, once a model has been created with acceptable surface accuracy using triangular facets, no additional refinement is needed to provide resolution to the predicted solidification pattern. Regularly gridded FDM models must consider both the required resolution of the temperature distribution and the surface accuracy of the model. Since point, edge and plane hot spots always lie on Voronoi vertices, edges, and faces, respectively, the Voronoi diagram exactly outlines the location and geometry of the hot spots. Additionally, since the geometry based approach is geometry based and non-iterative, instabilities do no arise due to an excessive time step size. The Voronoi diagram provides a unique skeletal representation which can be used to exactly reproduce the faceted solid model. FDM grids can only reproduce a corresponding solid model to within the resolution of the grid. Finally, the Voronoi diagram of a faceted model can have at most n^2 faces, where n is the number of facets. The actual number of Voronoi faces can be shown to be closer to a multiple of n faces. As a result, the Voronoi diagram representation should require a much smaller data set than a dense FDM grid.

6.3 Obstacles Inherent in Using the Voronoi Diagram of a Faceted Solid Model

Existing research has not produced algorithms for computing the exact Voronoi diagrams of general faceted polyhedra at this time. However, Voronoi diagrams are an extensive area of ongoing research in computational geometry, so it is reasonable to assume that the major issues concerning faceted solid model Voronoi diagram construction will be resolved shortly. This thesis identifies two such major issues: robust intersections and sensitivity to surface geometry. These issues become apparent when considering spherical geometries. By approximating a convex, symmetric and exact sphere with a non-convex, nearly symmetric, nearly regular faceted approximation of a sphere, the Voronoi diagram produced by this approximated model could lose symmetry and show intersection errors due to floating point inaccuracy. As a result, a Voronoi diagram which should consist of a single point in the center of the sphere could consist of a set of planes loosely grouped around its center [Figure 5.5]. Chapter Three gives evidence that both issues are being addressed. Sugihara [Sugihara94] has performed work on robust intersections of halfspaces, Ogniewicz and Ilg [Ogniewicz92] use residual functions to reduce the sensitivity of medial axis transformations to boundary geometry, and Sherbrooke et al. [Sherbrooke95] use an incremental advancement of a differential equation to reduce the sensitivity of their algorithm to surface geometry.

Problems concerning sensitivity to surface geometry are compounded by the use of the .STL file format. While this format allows the Voronoi diagram generator to accept files from a large number of commercial CAD systems, it does not maintain the original topology of the solid models developed on these packages. Therefore, as was the case with the SDRC I-DEAS Masters Series 2.0 .STL file generator used for this thesis, symmetry and convexity were not always preserved. Since the purpose of using the .STL file format is to allow for multiple sources of CAD models, the sensitivity issue cannot be resolved by enhancing a single .STL file generator to preserve topological characteristics. Instead, in order for robust Voronoi diagrams to be generated from .STL formatted input, the generation algorithm must be improved to reduce the Voronoi diagram's sensitivity to convexity and symmetry.

6.4 Suggestions for Further Study

Since the medial surface transformation has been established as a useful rapid solidification modeling tool, the first continuing research item that presents itself is the development of a robust, rapid application for calculating the Voronoi diagram for a general polyhedral model, and extracting the medial surface from the Voronoi diagram. The major obstacles to this task, as discussed in Section 6.3, are current research issues. Once a quality data structure and algorithm is implemented for this task, the application can be integrated into existing or new automated casting design applications. Specifically, the data structure can be used as an input for casting design expert systems, or as an input to a finite element mesh generator. Additionally, the Voronoi diagram can be gridded itself in order to produce the potential for discretized solutions on a smaller, yet more physically significant set of nodes.

In order to create a commercial level application which uses Voronoi diagrams for casting design, it must be combined with a more extensive analysis method. This is because, although the greatest included sphere method provides a basic approximation of solidification patterns, and accurately locates casting hot spots, it cannot offer the capability to handle non-uniform boundary conditions, mold filling considerations, placement of chills and insulation, and other part specific design issues. However, these issues can be resolved by a system that use the medial surface transformation first as a basis for rapid design decisions, and then as a basis for more detailed analysis and design decisions. Because the medial surface transformation is a purely geometric data structure from which heat transfer information can be directly extracted, the design automation approach can either accept the geometric information presented in the data structure and combine it with rules in a knowledge-based engineering approach or refine the geometry in order to make a mesh for a comprehensive heat transfer analysis approach. Because the medial surface transformation's completely and concisely represents a model's geometry in a compact manner, and because it provides a straightforward means of extracting the features of the geometry which most significantly effect heat transfer, the medial surface provides a good tool for use in automated casting design regardless of the approach.

References

[Beffel89]	Beffel M.J., Yu, K.O., Robinson, M., and Schneider, K.R., "Computer Simulation of Investment Casting Processes," *Journal of Metallurgy,* vol. 41, no. 2, February 1989, pp. 27-30.
[Blum67]	Blum, H., "A Method for Extracting New Descriptors of Shape", *Proceedings of the Symposium on Models for the Perception of Speech and Visual Form,* W. Whaten-Dunn, ed., MIT Press, Cambridge, Massachusetts, pp. 362-380.
[Bøhn93]	Bøhn, J.H., *Automatic CAD Model Repair,* Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, New York, August 1993.
[Brown93]	Brown, S.G., and Spittle, J.A., "A Rapid Alternative to Solidification Modeling," *Modern Casting,* vol. 83, no. 12, December 1993, pp. 24-25.
[Bose93]	Bose, N.K., and Garga, A.K., "Neural Network Design Using Voronoi Diagrams," *IEEE Transactions on Neural Networks,* vol. 4 no. 5, September 1993, pp. 778-787.
[Choset94]	Choset, H., and Burdick, J., "Sensor Based Planning and Nonsmooth Analysis," *Proceedings, IEEE Conference on Robotics and Automation,* pt. 4, 1994, pp. 3034-3041.
[Cruz95]	Cruz, M.E., and Patera, A.T., "A Parallel Monte-Carlo Finite Element Procedure for the Analysis of Multicomponent Random Media," *International Journal for Numerical Methods in Engineering,* vol. 38, 1995, pp. 1087-1121.
[Chvorinov40]	Chvorinov, N., "Theory of Solidification of Castings,"*Geissere,i* vol. 27, no. 10, May 1940, pp. 177-186.
[De Jonghe89]	De Jonghe, L.C., Chu, M.-Y., and Lin, M.K.F. "Pore Size Distribution, Grain Growth, and the Sintering Stress," *Journal of Materials Science,* vol. 24, 1989, pp. 4403-4408.
[DeKalb87]	DeKalb, S.W., Heine, R.W., and Uicker, J.J., "Geometric Modeling of Progressive Solidification and Casting Alloy Macrostructure," *AFS Transactions,* v. 95, 1987, pp. 281-294.

[Dobkin92] Dobkin, D.P., "Computational Geometry and Computer Graphics," *Proceedings of the IEEE,* vol. 80, no. 9, September 1992, pp. 1400-1411.

[Durkin94] Durkin, J., *Expert System Design and Development,* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1994.

[Estrin94] Estrin, L., "A Deeper Look at Solidification Software," *Modern Casting,* vol. 84, no. 7, July 1994, pp. 20-23.

[Ghosh94] Ghosh, S., and Mallett, R.L., "Voronoi Cell Finite Elements," *Computers and Structures,* vol. 50, no. 1, 1994, pp. 33-46.

[Ghosh95a] Ghosh S., Lee, K., and Moorthy, S., "Multiple Scale Analysis of Heterogeneous Elastic Structures Using Homogenization Theory and Voronoi Cell Finite Element Method," *International Journal of Solids and Structures,* vol. 32, no. 1, 1995, pp. 27-62.

[Ghosh95b] Ghosh, S., and Li, Y., "Voronoi Cell Finite Element Model Based on Micropolar Theory of Thermoelasticity for Heterogeneous Materials," *International Journal for Numerical Methods in Engineering,* vol. 38, 1995, pp. 1361-1398.

[Hill91] Hill, J.L., Piwonka, T.S., Berry, J.T., and Guleyopoglu, S., "Gating Design, Expert Systems, and Personal Computers... A Combination Worth Watching," *Incast,* vol. 4, no. 11, November 1991, pp. 8-12.

[Hjálmarrson94] Hjálmarrson, H., Sudhalkar, A., Gürsöz, L., and Prinz, F., "Automated Model Building for Moldability Analysis," *Design Engineering Division,* vol. 74, ASME, 1994. pp.71-85.

[ICI68] *The Investment Casting Handbook,* G.X. Diamond, ed., Investment Casting Institute, Chicago, Illinois, 1968.

[Jordan88] Jordan, C., Hill, J.L. and Piwonka, T.S., "Computer Designed Gating Systems: Promises and Problems," *AFS Transactions,* vol. 96, 1988, pp. 603-610.

[Kannan90] Kannan, K.S, Mahusudana, K., Venkataramani, R., Ganesh, N., and Prabhakar, O., "Modeling of Solidification Processes," *Indian Journal of Technology,* vol. 28, June-August 1990, pp. 460-474.

[Karsay72] Karsay, S., *Gating and Risering of Ductile Iron Castings*, Ferrous Foundry Consulting Co., New York, New York, 1972.

[Kotschi89] Kotschi, R.M., "The Missing Algorithms to Fully Computerize Gating, Risering, and CAM Tooling Manufacture, Part 1," *AFS Transactions,* vol. 97, 1989, pp. 689-694.

[Kreith80] Kreith, F., and Black, W.Z., *Basic Heat Transfer*, Harper and Row, New York, New York, 1980.

[Krozel90] Krozel, J., and Andrisani D., "Navigation Path Planning for Autonomous Aircraft: Voronoi Diagram Approach," *Journal of Guidance, Control and Dynamics,* vol. 13, no. 6, November/ December 1990, pp. 1152-1154.

[Lam92] Lam, L., Lee, S.-W., and Suen, C.Y., "Thinning Methodologies-- A Comprehensive Survey," *Transactions of Pattern Analysis and Machine Intelligence,* vol. PAMI-14, no. 9, September 1992, pp. 869-885.

[Lee82] Lee, D.T., "Medial Axis Transformation of a Planar Shape," *Transactions of Pattern Analysis and Machine Intelligence,* vol. PAMI-4, no. 4, July 1982, pp. 363-369.

[Morris89] Morris, R., and Smyrl, W., "Galvanic Interactions on Random Heterogeneous Surfaces," *Journal of the Electrochemical Society,* vol. 136, no.11, November 1989, pp. 3237-3248.

[Nieses87] Nieses, S.J., Uicker, J.J., and Heine, R.W., "Geometric Modeling of Directional Solidification Based on Section Modulus," *AFS Transactions,* vol. 95, 1987, pp. 25-30.

[Ogniewicz92] Ogniewicz, R., and Ilg, M., "Voronoi Skeletons: Theory and Applications," *Proceedings, 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* June 1992, pp. 63-69.

[O'Rourke79] O'Rourke, J., and Badler, N., "Decomposition of Three Dimensional Objects into Spheres," *Transactions of Pattern Analysis and Machine Intelligence,* vol. PAMI-1 no. 4, July 1979, pp. 295-305.

[O'Rourke94]	O'Rourke, J., *Computational Geometry in C,* Cambridge University Press, New York, New York, 1994.
[Palagi94]	Palagi, C.L., and Aziz, K., "Modeling Vertical and Horizontal Wells with Voronoi Grid," *SPE Reservoir Engineering,* vol. 9 no. 1, February 1994, pp. 15-21.
[Pei87]	Pei, Q.X., Bai, T.S., and Liu, P.C., "Riserless Design of Ductile Iron Castings by Computer Program," *AFS Transactions,* vol. 95, 1987, pp. 443-450.
[Pehlke88]	Pehlke, R.D., "Heat Flow Analyses for Solidification and Cooling-State of the Art," *Modeling of Casting and Welding Processes IV,* A.F. Gamei and G.J. Abbaschian, eds., TMS 1988, pp. 3-13.
[Peleg81]	Peleg, S., and Rosenfeld, A., "A Min-Max Medial Axis Transformation," *Transactions of Pattern Analysis and Machine Intelligence,* vol. PAMI-3 no. 2, March 1981, pp. 208-10.
[Preparata77]	Preparata, F.P., "The Medial Axis of a Simple Polygon," *Proceedings, 6th Symposium on the Mathematical Foundations of Computer Science,* September 15-17, 1977, pp. 443-450.
[Preparata85]	Preparata, F.P., and Shamos, M.I., *Computational Geometry,* Springer-Verlag, New York, New York, 1985.
[Rolland92]	Rolland, F., Chassery, J.-M., and Montanvert, A., "3D Medial Surfaces and Skeletons," *Visual Form: Analysis and Recognition*, C. Arcelli, L.P. Cordella, and G.S. di Baja, eds., Plenum Press, New York, New York, 1992, pp. 443-450.
[Samet83]	Samet, H., "A Quadtree Medial Axis Transform," *Communications of the ACM,* vol. 26, no. 9, September 1983, pp. 680-693.
[Sandia92]	"Sandia Laboratories FASTCAST Program Being Developed to Assist in All Phases of Investment Casting Design and Production," *Incast,* vol. 5 no. 11, November 1992, pp. 8-11.
[Sherbrooke95]	Sherbrooke, E.C., Patrikalakis, N.M., and Brisson, E., "Computation of the Medial Axis Transform of 3D Polyhedra," *Proceedings, 3rd Symposium on Solid Modeling and Applications,* Salt Lake City, Utah, May 17-19, 1995, pp. 187-199.

[Shirriff93] Shirriff, K., "Generating Fractals from Voronoi Diagrams," *Computers and Graphics,* vol. 17, no. 2, 1993, pp. 165-167.

[Sirilertworakul93] Sirilertworakul, N., Webster, P.D., and Dean, T.A., "Computer Prediction of Location of Heat Centres in Castings," *Material Science and Technology,* vol. 9 no. 10, October 1993, pp. 923-928.

[Stifter91] Stifter, S., "An Axiomatic Approach to Voronoi Diagrams in 3-D," *Journal of Computer and System Sciences,* vol. 43 no. 2, October 1991, pp. 361-379.

[Sudhalkar93] Sudhalkar, A., Gürsöz, L., and Prinz, F., "Continuous Skeletons of Discrete Objects," *Proceedings, 2nd Symposium on Solid Modeling and Applications,* Montreal Canada, May 13-15. pp. 85-94.

[Sugihara92] Sugihara, K.,and Iri, M., "Construction of the Voronoi Diagram for 'One Million' Generators in Single Precision Arithmetic," *Proceedings of the IEEE,* 1992, pp. 1471-1484.

[Sugihara93] Sugihara, K., "Approximation of Generalized Voronoi Diagrams by Ordinary Voronoi Diagrams," *CVGIP: Graphical Models and Image Processing,* vol. 55 no. 6, 1993, pp. 522-531.

[Sugihara94] Sugihara, K., "A Robust and Consistent Algorithm for Intersecting Convex Polyhedra," *Eurographics `94,* M. Dæhlen and L. Kjelldahl, eds., Oslo, Norway, September 12-16, 1994, pp. 45-54.

[Takata94] Takata, S., and Tsai, M.-D., "Model Based NC Programming for End Milling Operations," *PED v.68-2, Manufacturing Science and Engineering,* vol. 2, ASME 1994, pp. 809-818.

[Taniguchi91] Taniguchi, N., and Kobayashi, T., "Finite Volume Method on the Unstructured Grid System," *Computers and Fluids,* vol. 19 no. 3/4, 1991, pp. 287-295.

[Tanimoto92] H. Tanimoto and N. Shigyo, "Discretization Error in MOSFET Device Simulation," *IEEE Transactions on Computer Aided Design,* vol. 11 no. 7, July 1992, pp. 921-925.

[Tei-Ohkawa94] Tei-Ohkawa, T., Edagawa, K., Takeuchi, S., and Masuda-Jindo, K., "Atomic Packing Geometries and Lattice Properties of a Model Decagonal Phase," *Materials Science and Engineering A:*

Structural Materials, Properties, Microstructure, and Processing, vol. 181-182, pt. 2, May 15, 1994, pp. 833-836.

[Tu93] Tu, J.S., Olinger, D.M., and Hines, A.M., "Computer Aided Development of an Investment Casting Process," *Journal of Metallurgy,* vol. 45 no. 10, October 1993, pp. 29-32.

[Upadhya93] Upadhya, G., and Paul, A.J., "Rational Design of Gating and Risering For Castings: A New Approach Using Knowledge Base and Geometric Analysis," *AFS Transactions,* vol. 101, 1993, pp. 919-925.

[Upadhya94] Upadhya, G., and Paul, A.J., "Solidification Modeling: A Phenomonenological Review," *AFS Transactions,* vol. 102, 1994, pp. 69-80.

[Wukovich89] Wukovich, N., and Metevelis, G., "Gating: The Foundryman's Dilemma, or Fifty Years and Still Asking 'How?'," *AFS Transactions,* vol. 97, 1989, pp. 285-302.

[Zhang94] Zhang, H.G., Webster, P.D., and Dean, T.A., "Computer Aided Design of Feeders for Castings," *Journal of Engineering Manufacture,* Part B, vol. B4, 1994, pp. 279-87.

Vita

Scott Houser was born in Steven's Point, Wisconsin on January 25, 1969. He moved with his family to State College, Pennsylvania, and then on to Westminster, Maryland, where he attended Westminster High School. He obtained a Bachelor of Science degree at Cornell University's Sibley School of Mechanical and Aerospace Engineering, and interned at Dresser Rand, Inc., Painted Post, New York as an undergraduate. Following graduation from Cornell, Scott worked for over two years for Westinghouse's Bettis Atomic Power Laboratory division. While working for Bettis, he graduated from the Naval Nuclear Power School in Orlando, Florida, and qualified as an instructor and Engineering Officer of the Watch on a prototype reactor at the Nuclear Power Training Unit of the Charleston Naval Weapons Station, Charleston, SC.

An interest in engineering design and concurrent engineering prompted Scott's return to academia. He elected to attend the Virginia Polytechnic Institute and State University in Blacksburg, Virginia, where his research interests included computer aided engineering and computational geometry. He is currently employed with Phoenix Integration, headquartered in the Virginia Tech Corporate Research Center in Blacksburg Virginia, as a mechanical and software engineer.

[illegible]

Scott A. Houser