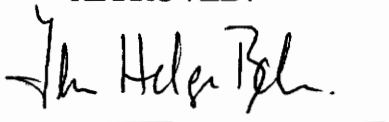
ADAPTIVE HIGH-PRECISION EXTERIOR, HIGH-SPEED INTERIOR, LAYERED MANUFACTURING

by

Emmanuel Sabourin

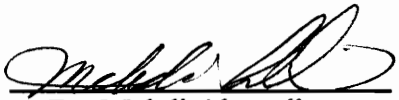
Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fufillment of the requirements for the degree of
MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

APPROVED:



Dr. Jan Helge Bøhn, Chairman

Dr. Ronald G. Kander



Dr. Mehdi Ahmadian

February 1996
Blacksburg, Virginia

ADAPTIVE HIGH-PRECISION EXTERIOR, HIGH-SPEED INTERIOR LAYERED MANUFACTURING

by

Emmanuel Sabourin

Jan Helge Bøhn, Chairman

Department of Mechanical Engineering

(ABSTRACT)

Contemporary layered manufacturing systems build parts using a constant layer thickness. Such systems must seek a compromise between fast fabrication and large inaccuracies on the one hand, and slow fabrication and high precision on the other. This thesis demonstrates how this compromise can be avoided. Specifically, it segregates solid models described in the .STL file format into exterior and interior regions. The exterior regions are fabricated with thin, dense, adaptive thickness layers, using narrow material deposition, to ensure high-precision part surfaces. Concurrently, the interior regions are fabricated with thick, sparse layers, using wide material deposition, to maximize build speed and minimize material usage. Experimental software has been developed and sample parts have been fabricated to demonstrate proof of concept.

ACKNOWLEDGMENTS

I would like to thank...

my parents and my friends for their unending support and confidence;

my girlfriend, Fanny, for her incredible support in every day life;

Dr. J.H. Bøhn, my advisor, for opening to me the world of CAD/CAM and rapid prototyping;

whoever came up with the C++ programming language.

Finally, endless thanks to Scott Houser for helping me in putting together this thesis, for making my English understandable, and, above all, lightening up long work days with incredible jokes.

This research was supported by the Naval Surface Warfare Center, Dahlgren Division under contract N60921-89-D-A239, Order 0045. Any opinions, findings, conclusions, or recommendations expressed in this thesis are those of the author and do not necessarily reflect the views of the Naval Surface Warfare Center, Dahlgren Division.

Contents

List of figures

List of tables

Chapter 1

Introduction

Physical rapid prototyping, otherwise known as automated fabrication or solid freeform fabrication, is a modern family of technologies that generate three-dimensional, solid objects under computer control. One important advantage of rapid prototyping over traditional fabrication processes is *Solid WYSIWYG* ("What-you-see-is-what-you-get"). These technologies give design and production engineers two advantages: (1) the freedom to generate previously unproducible shapes, and (2) the ability to generate conventionally producible shapes with more speed, accuracy and cost effectiveness than by traditional manufacturing methods. Rapid prototyping research seeks to enhance these advantages by focusing upon improvement of fabrication speed, part dimensional accuracy, and material selection. This thesis focuses on improving the fabrication speed and the part accuracy.

Physical rapid prototyping processes include *subtractive* processes, which start with a block of solid material and remove material to achieve the desired shape; *additive* processes, which apply material in successive layers to yield a complete part; and *formative* processes, which apply mechanical forces to a material to form it into the desired shape. Automated subtractive processes, which include computer numerical control (CNC and DNC) machining, and formative processes, such as stamping and forming, are well established and understood rapid prototyping processes. Additive rapid prototyping processes have emerged more recently, beginning with the commercial introduction of the Stereolithography Apparatus (SLA) by 3D Systems, Inc. in 1987. An abundance of additive processes have been introduced since that time.

In essence, an additive fabricator is the 3D analog of a computer printer. The geometry of the part to be fabricated can originate from three sources: computer aided design (CAD), reverse engineering (e.g., computer tomography data acquisition), or formal mathematics (e.g., surface equations) [Burns93]. In most cases, the geometry representation taken from these sources cannot drive the fabricator directly. Typically, the geometry must be preprocessed; it must be converted into a set of parallel, horizontal slices of a desired thickness, from which the appropriate numerical control (NC) code that drives the fabricator can be derived.

A number of observations concerning this preprocessing can be made:

- The positional data accuracy, both hardware and software, within each cross section (transversal dimension) is as good as, or better than, the width of the material deposition device; typically being a laser beam or a nozzle.
- Because the part is built of slices having a specific thickness, non-horizontal part surfaces tend to be somewhat rough; specifically, there is often a noticeable stair-stepping effect between consecutive slices of the part.
- The stair-stepping effect decreases with reduced slice thickness. Thus the thinner the slices, the more accurate the part becomes.
- Given that a thick slice and a thin slice have similar build times, reducing the number of slices that are required will reduce overall build time.

Consequently, the rapid prototyping industry is currently faced with the dilemma of choosing between part accuracy (thin slices) and build speed (thick slices). Ideally, those using rapid prototyping to produce functional parts want both accurate and fast fabrication.

1.1 Problem statement

The goal of this thesis is to overcome the dilemma facing the rapid prototyping industry and let the user attain enhanced part accuracy without increasing the build time. In particular, it will demonstrate a method in which the surface of the part is built with thin slices to yield maximal surface accuracy, while the interior of the part is built with thick slices to minimize overall build time. This approach should be suitable for several layered fabrication processes, though this research will focus primarily on Fused Deposition Modeling (FDM).

1.2 Solution outline

In typical additive fabrication processes, each slice represents a cross section of the model to be fabricated. Global accuracy is predominately proportional to the number of slices. The stair-stepping effect is intrinsic to the slicing process (Fig. 1.1), and is therefore an ever-present source of dimensional inaccuracy.

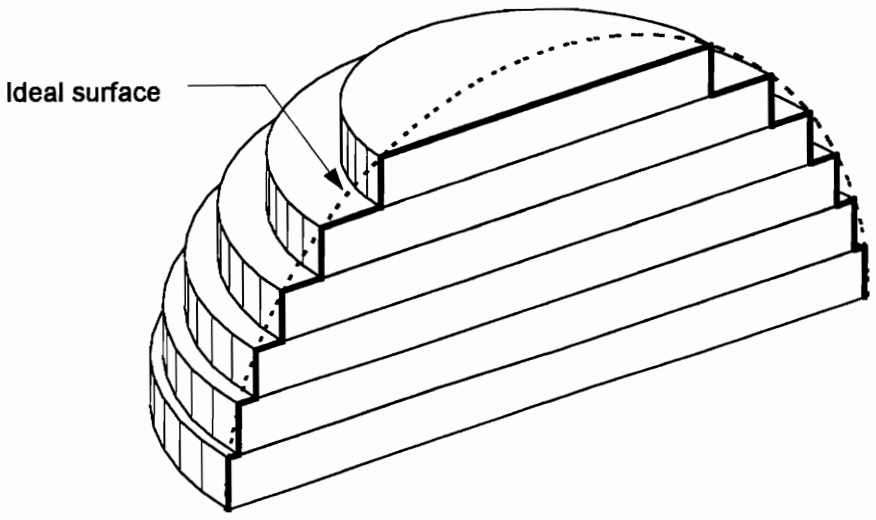


Figure 1.1: **Excessive stair-stepping inaccuracy due to standard uniform slicing**

This inaccuracy can be minimized without increasing build time by segregating the interior of each slice from its contour: the part is built by first creating an external shell of thin slice contour layers, followed by filling the part with a thicker interior layer (Fig. 1.2). This method simultaneously minimizes stair-stepping inaccuracies by adapting the thickness of the thin exterior slices to better match the part's surface geometry, and minimizes overall fabrication time by building the thick internal layers as open lattice structures. The lattices minimize material usage and overall build time without significantly affecting the structural integrity of the part.

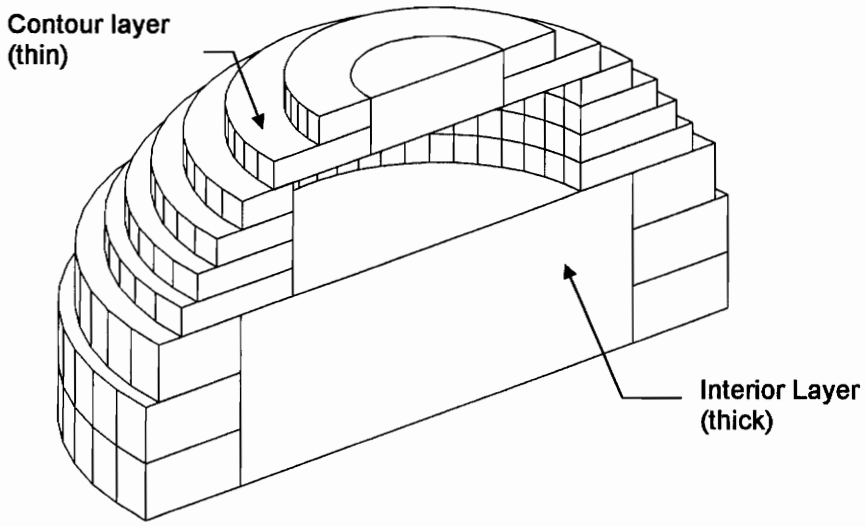


Figure 1.2 : **Adaptive slicing of contour and interior regions. By increasing the number of contour layers, surface accuracy is improved and stair-stepping is decreased. By increasing the thickness of the internal layers, overall fabrication time is reduced (Note : the middle interior layer is not shown for clarity).**

To achieve the goal presented above, the model is first sliced into thick slices with the maximum thickness allowed by the fabricator (Fig. 1.3). The external contours of these slices are then offset into the slice interiors to define the interior and exterior regions of the slice (Fig. 1.4). Finally, the exterior regions are sliced adaptively into still thinner slices, as needed, based on the vertical slope of the part surface along the thick slice

contours, to ensure sufficient external surface smoothness. As Chapter 4 shows, this is both a simple and reasonable method for creating a model with a fast to produce interior and an accurate exterior.

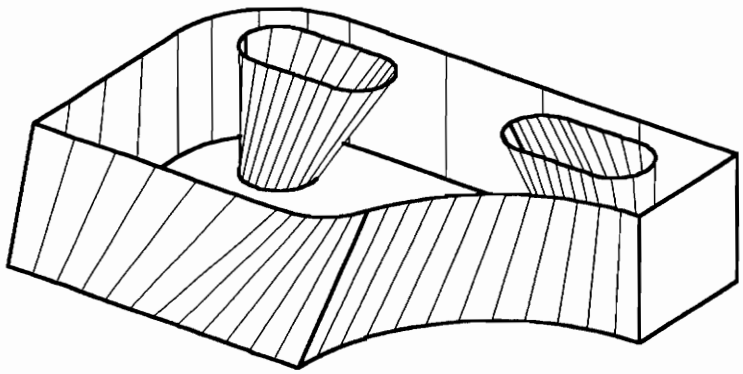


Figure 1.3: **A thick slice of a part as produced using a conventional slicing method.**

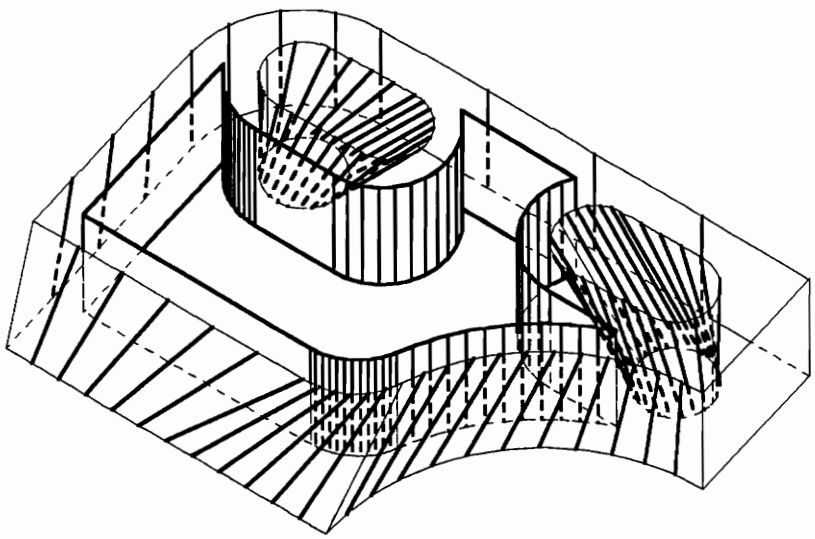


Figure 1.4: **Same slice subdivided into interior and exterior regions. The external regions is in turn sliced into thinner slices in order to maintain surface accuracy.**

1.3 Scope of research and thesis organization

The goal of this thesis is to demonstrate how a closed physical part can be built with thin contour slices and thick internal slices. The remainder of this thesis is divided into the following chapters:

Chapter 2 presents the background material needed for achieving adaptive layered manufacturing, which include terminology, mathematical bases, applicability of the thesis to the available rapid prototyping technologies, and a discussion on the .STL file format.

Chapter 3 is a literature survey on the current work that relates to adaptive layered manufacturing; in particular, the current effort to improve the file formats, to speed up the slicing process, and to increase fabrication accuracy.

Chapter 4 shows in detail the methods developed in this thesis to achieve adaptive high-precision exterior, high-speed interior layered manufacturing.

Chapters 5 presents performance and results of adaptive high-precision exterior, high-speed interior layered manufacturing and compares them to existing layered fabrication technologies.

Chapter 6 discusses the results, draws conclusions, and gives recommendations for future work.

Chapter 2

Background Material

This chapter presents the mathematical and rapid prototyping concepts used in this research. Section 2.1 discusses four topics required to understand the slicing and tool path generation procedures: basic geometric definitions, edge/plane intersections, edge/line intersections, and contour orientation. Section 2.2 discusses concepts used in making these procedures faster and more robust: topology and uniform spatial partitioning. Section 2.3 discusses the basic algorithms used in this thesis: marching algorithms, contour offsets, and the odd winding rule. Section 2.4 presents the applicability of this work to commercially available rapid prototyping processes, and section 2.5 provides a brief description of the .STL file format.

2.1 Mathematical concepts

2.1.1 Basic geometric definitions

Vertex A vertex corresponds to a zero-dimensional entity (a point) in 3D space. It is described by three spatial coordinates [Bøhn93].

Edge An edge is a one-dimensional entity connecting two vertices [Bøhn93]. Often, an edge has a direction; in this case one of its vertices is called its head, and the other is called its tail.

Facet A facet is a triangular planar surface delimited by three edges. It is described explicitly by three vertices; the edges that define a facet are implied by the three vertices. A facet has an orientation, and, in particular,

a directed normal. This normal is perpendicular to the facet plane and oriented toward the exterior of the model.

Solid A solid is a collection of facets. It is closed and properly oriented.

Orientation Solid models have a material and a non-material side. The orientation of each facet describing solid models must be known in order to properly separate the interior from the exterior. The orientation of a facet can be explicit, e.g., using a surface normal, or implied from the direction of the loop formed by the three directed edges (using the right-hand rule) [Bøhn93].

Slice A slice is a two-dimensional entity (a plane) that represents a horizontal cross section of a solid model.

Contour A contour is a 2D continuous collection of directed edges that bounds a slice. It is closed and oriented.

Layer A layer is a subvolume of a model that lies between two slices. It is a mathematical representation of what is fabricated.

2.1.2 Edge/Plane intersections

This section presents methods for computing edge/plane intersections, which are fundamental to the slicing procedure. The following section presents edge/line intersections, which are fundamental to tool path generation. It is important that both these operations are designed to be as fast as possible because they are used extensively, and repetitively, in generating slices and tool paths. The methods discussed below are intended to optimize the floating point operations.

The slicing planes used to slice a solid model are by definition horizontal. Rock and Wozny [Rock91b] show how this greatly simplifies the edge/plane intersection

process. If one edge vertex is above the plane and the other is below, then the edge intersects the plane.

In the Euclidean coordinate system $(\vec{i},\vec{j},\vec{k})$, the equation of a horizontal plane at a given height h is given by:

$$z = h \tag{2.1}$$

If the edge is defined by two vertices $\mathbf{p}=(x_p, y_p, z_p)$ and $\mathbf{q}=(x_q, y_q, z_q)$ as described by the parametrization

$$\mathbf{P}(t) = \mathbf{p} + t\,(\mathbf{q}-\mathbf{p}) \tag{2.2}$$

then the edge/plane intersection can be found at [Rock91b]

$$t = \frac{h - z_p}{z_q - z_p} \tag{2.3}$$

Assuming one floating point operation (FLOP) for each addition, subtraction, and multiplication, respectively, and five FLOPS for each division [Bøhn89], t can be computed in seven FLOPS and the (x, y) coordinates of the intersection in another six FLOPS, for a total of 13 FLOPS.

2.1.3 Edge/Line intersections

Edge/edge and edge/line intersections are necessary during tool path generation, and in particular, when offsetting contours and rastering the interior.

The problem of intersecting an edge E with a line L, both being in the plane Π, can be transformed into an edge/plane intersection by regarding the line L as the intersection of the plane Π and as perpendicular plane P (Fig. 2.1). The plane P can be defined by any point $\mathbf{p}=(x_p, y_p, z_p)$ on L, and any vector $\vec{N}=[n_x, n_y, n_z]$ perpendicular to L in Π:

$$P:\ n_x(x_p - x) + n_y(y_p - y) + n_z(z_p - z) = 0 \tag{2.4}$$

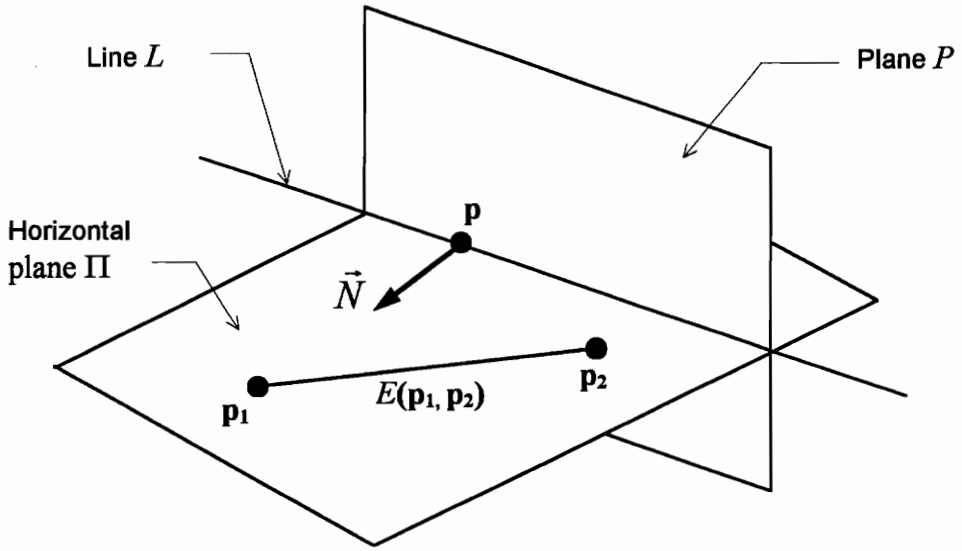


Figure 2.1: **Edge/Line intersection**

Since Π is a horizontal plane, $n_z = 0$. Hence, the function

$$f(x, y) = n_x(x_p - x) + n_y(y_p - y) \tag{2.5}$$

can be defined with the sign of f determining which side of the plane P the point (x, y) lies. An edge $E(\mathbf{p}_1, \mathbf{p}_2)$ in Π will therefore intersect the line L if and only if

$$f(\mathbf{p}_1) f(\mathbf{p}_2) < 0 \tag{2.6}$$

The normal $\vec{N}$ can be defined given two distinct points, $\mathbf{L}_1 = (x_{L1}, y_{L1}, z_{L1})$ and $\mathbf{L}_2 = (x_{L2}, y_{L2}, z_{L2})$, on L

$$\vec{N} = [(y_{L2} - x_{L2}), (y_{L1} - x_{L1}), 0] \tag{2.7}$$

for a computational cost of 2 FLOPS. The cost, therefore, of determining if an edge E intersects a line L, both in the horizontal plane Π, is 12 FLOPS.

To determine if two edges lying in the horizontal plane Π intersect, each edge must be tested against the infinite line of the other edge. The computational cost in this case is therefore 24 FLOPS.

The coordinates of the intersection between an edge $E(\mathbf{p}_1,\mathbf{p}_2)$ and a line L, or another edge on L, in the horizontal plane Π, can be found by projection of the vectors $(\mathbf{p}_1 - \mathbf{p})$ and $(\mathbf{p}_1 - \mathbf{p}_2)$ onto $\vec{N}$, where $\vec{N}$ is a normal of L in Π and P in any point $\notin \{\mathbf{p}_1,\mathbf{p}_2\}$ on L. The ratio of these projections correspond to the parametric location of the intersection on the line of edge E (Fig. 2.2).

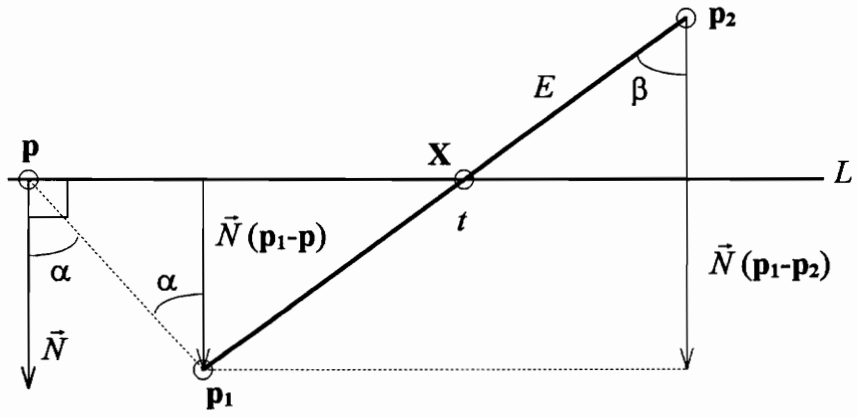


Figure 2.2: **Edge/Line intersection computation**

$$\mathrm{X} = \mathbf{p}_1 + t\,(\mathbf{p}_1 - \mathbf{p}_2) \quad \text{where} \quad t = \frac{\cos\alpha\|\mathbf{p}_1 - \mathbf{p}\|}{\cos\beta\|\mathbf{p}_1 - \mathbf{p}_2\|} \tag{2.8}$$

This can be expressed as

$$t = \frac{\left(\dfrac{\vec{N}(\mathbf{p}_1 - \mathbf{p})}{\|\vec{N}\|\|\mathbf{p}_1 - \mathbf{p}\|}\right)\|\mathbf{p}_1 - \mathbf{p}\|}{\left(\dfrac{\vec{N}(\mathbf{p}_1 - \mathbf{p}_2)}{\|\vec{N}\|\|\mathbf{p}_1 - \mathbf{p}_2\|}\right)\|\mathbf{p}_1 - \mathbf{p}_2\|}$$

or simply as

$$t = \frac{\vec{N}(\mathbf{p}_1 - \mathbf{p})}{\vec{N}(\mathbf{p}_1 - \mathbf{p}_2)}$$

and for computational purposes

$$t = \frac{n_x\left(x_{p1} - x_p\right) + n_y\left(y_{p1} - y_p\right)}{n_x\left(x_{p1} - x_{p2}\right) + n_y\left(y_{p1} - y_{p2}\right)} \tag{2.9}$$

Noting that parts of this computation have already been computed (Eq. 2.5), t can be further simplified to

$$t = \frac{-f(\mathbf{p}_1)}{n_x\left(x_{p1} - x_{p2}\right) + n_y\left(y_{p1} - y_{p2}\right)} \tag{2.10}$$

Hence, with the z coordinate known, the remaining intersection coordinates (Eq. 2.8) can therefore be found in 16 FLOPS when using Eq. 2.10.

2.1.4 Contour orientation convention

The material in each layer is bound by the set of contours on the corresponding slice. The convention for these contours is that they are directed such that the material always lies to the left of the contours, as viewed in the direction of the contour. Hence, external contours are directed counterclockwise (CCW), and internal contours are directed clockwise (CW) (Fig 2.3).

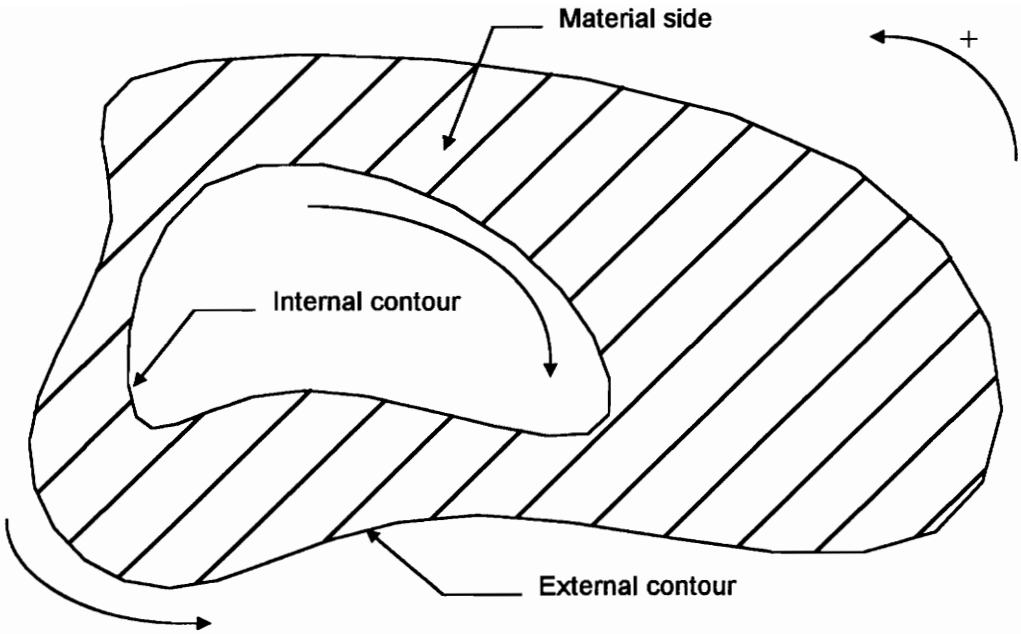


Figure 2.3: **Cross section of a solid model showing proper contour orientation.**

2.2 Important algorithm concepts

The subsequent sections present two important concepts that are extensively used throughout this thesis; namely, topology and spatial partitioning. The use of topology greatly enhances the navigation through the model search space, while the use of spatial partitioning significantly reduces the size of the search space.

2.2.1 Topology

Weiler defines topology as "a set of properties invariant under a specified set of geometric transformations" [Weiler86]. The specific topology used in this thesis is the connectivity, which is information which relates each component of a composite entity to its subcomponents, the components which contain it as a subcomponent, or its neighboring components. For example, connectivity for a triangular facet may include references to each of its three vertices; in turn, each vertex may include references to each facet to

which it belongs. Furthermore, a triangular facet may contain references to its neighboring facets, and a vertex may contain references to its neighboring vertices.

Topology, and especially connectivity, is usually used to facilitate efficient manipulation of geometric entities. This often includes avoiding scanning through an entire composite structure when looking for a particular entity. In the slicing procedure, for example, a set of triangular facets define the solid model to be sliced. A slicing plane cuts subsets of these facets, which, because they all intersect the slicing plane and are members of a closed solid model, are all connected through common edges. These subsets constitute the contours that bound a slice. With or without topology, finding the entire set of intersecting facets requires a check of every facet in the model. However, with topology, only one facet per contour needs to be found. The rest of the contour facets are easily found since adjacent facets are known to each facet. Without connectivity, each intersecting facet in a slice must be ordered relative to other intersecting facets to determine the shapes of the contours. This is a computationally expensive operation.

To give a computational cost approximation, assume m facets among a database of N triangular facets intersect a given slicing plane. Without topology, N facets are checked for intersection, which is an $O(N)$ operation. To generate a continuous contour, each of the m facets found must be checked against each other, which is an $O(m^2)$ operation; the total complexity of slicing operation is $O(N) + O(m^2)$. With topology, an $O(N)$ search is still required to find the m facets; however, since their connectivity is known, the generation of a continuous contour is $O(m)$. Therefore, with topology, the complexity is reduced to $O(N) + O(m)$.

Topology also provides great flexibility in the management of the entities. For example, an edge can be defined by two vertices. In terms of structure, the edge can be an entity that holds references to the vertices. The vertices are themselves entities that hold spatial coordinates. In this manner, if one of the extremities of the edge is to be changed,

only the coordinates of the relative node need to be changed. The edge is not redefined wholly: the edge entity remains exactly the same, only the characteristics of one of its vertices changes.

The example developed above does not include the time necessary to identify and organize the topology. The effective cost of identifying and organizing the topology is difficult to forecast. This difficulty, however, can in part be overcome if every entity is created initially with topological information.

2.2.2 Spatial partitioning

Spatial partitioning refers to the concept of splitting a geometric space into several distinct subspaces. For instance, a data set contained within the original space can be split into smaller sets which are contained in a corresponding subspace. Searching for an element inside these smaller subspaces is often much faster than browsing the entire space. This is because, although the complexity remains the same, e.g., $O(n^2)$, the effective value of n is decreased. For example, if the n data are evenly divided among m subspaces, an $O(n^2)$ algorithm will see an overall reduction in operations by a factor of $1/m$: $mO((n/m)^2) = O(n^2/m)$. Typically, the space is split into slabs or into boxes [Preparata88], often also referred to as buckets [Rock91b]. If the subspaces are of equal dimensions, then the partitioning is called uniform. Uniform spatial partitioning is popular because it is easy to implement while also being computationally extremely efficient [Franklin90, Preparata88].

Uniform spatial partitioning tends to lose its effectiveness, however, when the data set is not evenly distributed across the original space. The solution to this problem is non-trivial, one of which is binary spatial partitioning (BSP) [Naylor90]. There, the space is partitioned to fit the data; specifically, aiming to create buckets with similar amount of data.

2.3 Basic geometric algorithms

The subsequent sections present the basic geometric algorithms used in this thesis; namely, marching algorithms [Rock91b], contour offsetting [Yang94, Farouki94], and the odd-winding rule [Gaskins92]. Marching algorithms are used both to create slice contours and to recreate simple closed contours during tool path generation. A marching algorithm requires a valid geometric model with knowledge of connectivity between the components of the model to be marched through. The contour offsetting algorithm defines the boundary between the interior and exterior regions of a layer and generates contour tool paths. The odd-winding algorithm merges and classifies regions bounded by contours.

2.3.1 Marching algorithm for slicing

Model slicing and contour generation of a faceted solid model is greatly simplified if facet/facet connectivity is known. Rock and Wozny [Rock91b] demonstrated that given this connectivity, an ordered contour could be extracted given an initial facet that intersects the slice plane. They observed that each intersecting facet has two intersecting edges (Fig 2.4). The facet's contour edge is bounded by these two edges' intersection with the slice plane, and is connected with two contour edges arising from the intersection of the two other facets adjacent to the two intersecting edges. Hence, the contour edges can be found by marching from intersecting facet to intersecting facet.

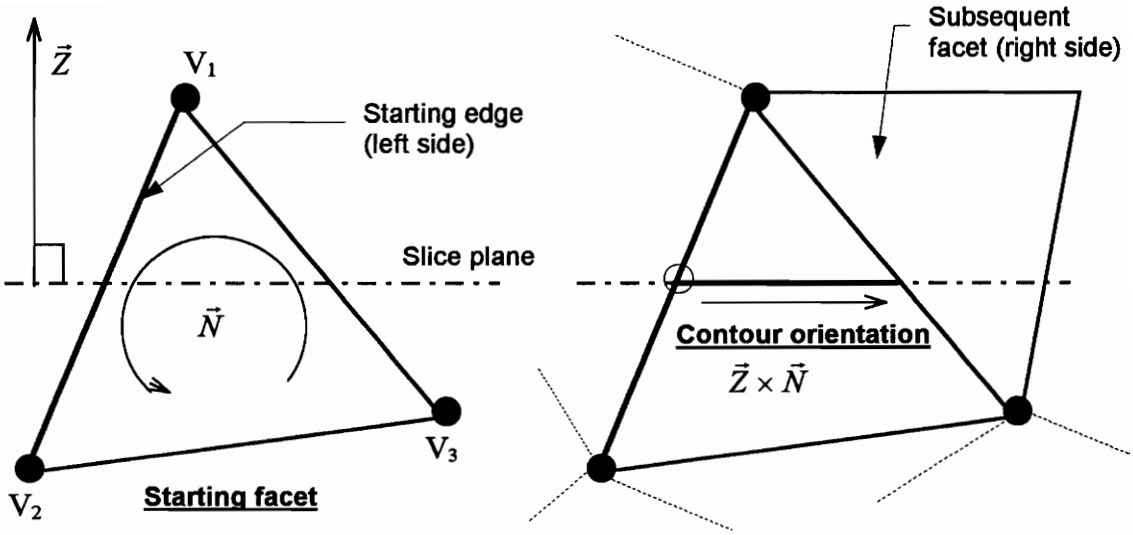


Figure 2.4: **The marching direction is given by $\vec{Z} \times \vec{N}$.**

The direction of marching is given by $\vec{Z} \times \vec{N}$, where $\vec{Z}$ is the vertical vector and $\vec{N}$ is the facet's surface normal. Rock and Wozny simplify this by observing that, for instance, a facet $f_a(V_1, V_2, V_3)$, with the vertex V_1 above and the vertex V_2 below the slice plane, the next facet will be f_{13} if V_3 is below the plane or f_{23} if V_3 is above the slice plane, respectively (Fig. 2.5).

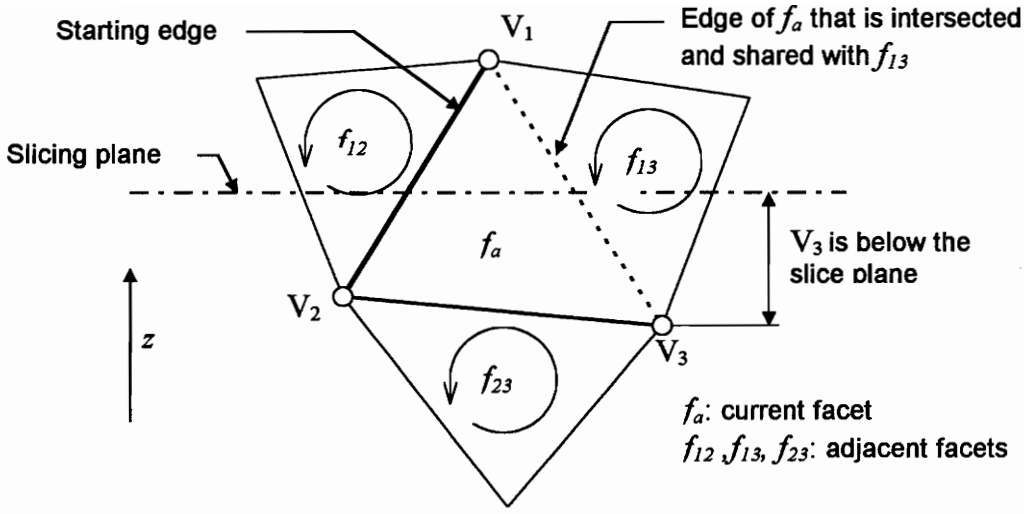


Figure 2.5: **Given a starting facet *fa* and a starting edge (v_1, v_2), the next facet to intersect is determined by the relative position of the vertex v_3 to the slicing plane. Here, since v_3 is below the plane, f_{13} is the next facet to be intersected by the algorithm.**

2.3.2 *Marching algorithm for contour reconstruction*

When generating a tool path, the contours of the slice are offset into the model. These contours may intersect themselves or other contours after they are offset. Since the purpose of performing these contour offsets is to separate the interior of the layer from the exterior in order to allow a sparse tool path on the interior and a dense tool path on the exterior, these contours cannot be allowed to intersect. If the contours generated to define these regions intersect, the regions themselves cannot be distinguished from each other. Therefore, after each slice contour is independently offset into the model, the set of contours must be modified to remove intersections and to ensure that the resulting contours are simple and closed. This intersection removal constitutes the first step towards constructing a distinct interior and exterior region; following this first step, the interior and exterior can be constructed by performing direction checking to remove contours which are invalid or not useful [Yang94]. Contour intersection and direction checking are discussed in chapter 4. This section discusses a marching algorithm which

uses the concepts employed by Yang [Yang94] to create simple, closed contours from a set of intersecting contours (Fig 2.6).

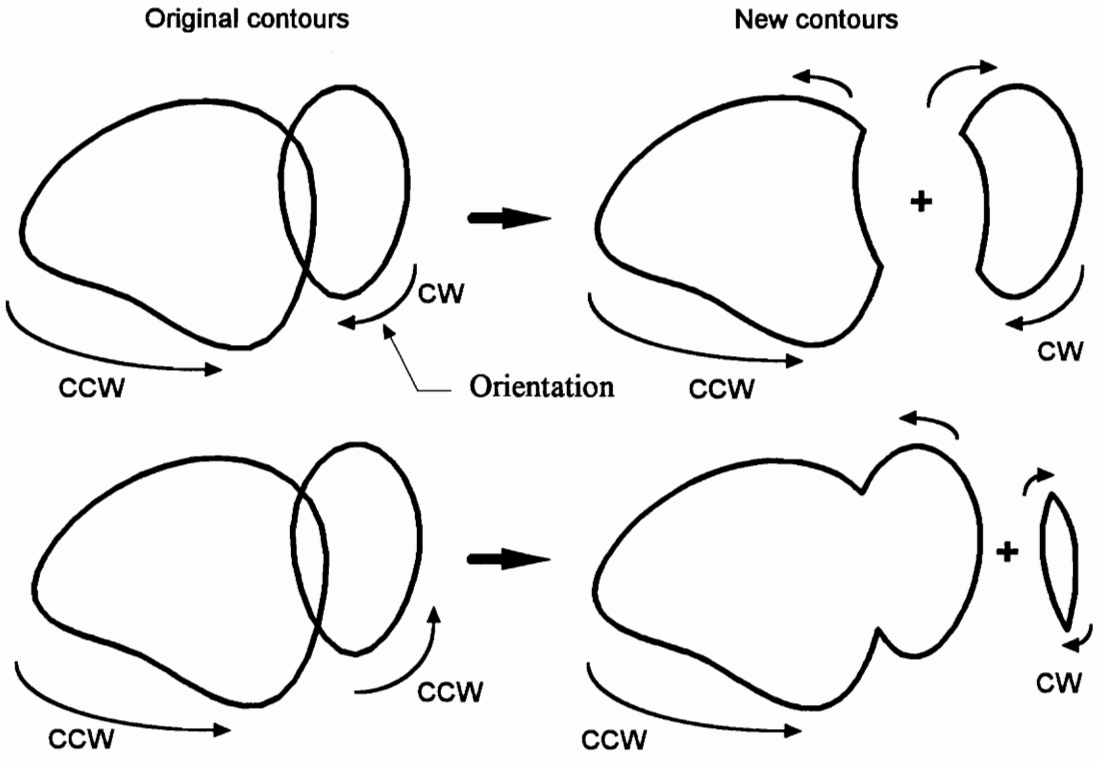
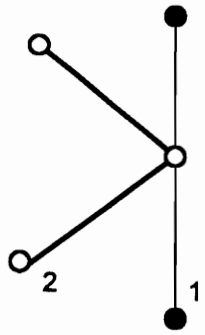


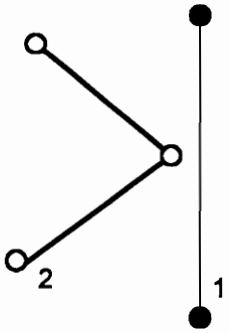
Figure 2.6: **Contour reconstruction after intersection.**

In order to properly create a simple, closed path through a set of intersecting contours, the types of intersections encountered must first be constrained to ensure that all intersections points can be navigated correctly with a simple set of rules. Since the contours generated in this thesis consist of connected path edges, the following scenarios may occur: exactly two edges may intersect at a point which does not coincide with either edge's vertices; an edge may intersect with the common vertex of two connected edges (Fig. 2.7(a)); and three or more edges may intersect at a common point (Fig. 2.8(a)). Combinations of the above scenarios may be avoided by perturbing an intersecting edge vertex and re-evaluating the intersection. Using this technique, an intersection between three or more edges can be simplified into a series of two-edge intersections (Fig. 2.8(b)),

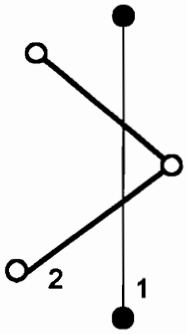
and an edge/vertex intersection can be simplified into: (1) an edge/edge intersections in which both edges adjacent to the affected vertex now intersect the third edge (Fig 2.7(a)), or (2) no intersection at all (Fig 2.7(b)).



(a) An edge of contour #1 (thin) intersects a vertex of contour #2 (thick).

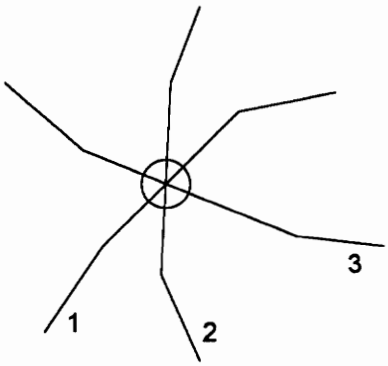


(b) The edge is perturbed leading to no intersection.

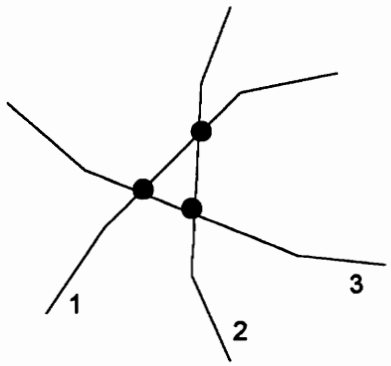


(c) The edge is perturbed leading to 2 regular edge/edge intersections.

Figure 2.7: **Exception handling for edge/vertex intersection.**



(a) Three contours intersect at the same point.



(b) The intersecting contour edges are slightly perturbed to cause three regular edge/edge intersections.

Figure 2.8: **Exception handling in multiple contour intersection.**

By ensuring that the only intersections encountered are simple edge/edge intersections, an edge to edge march around a contour path will result in two possible scenarios (Fig. 2.9). Case A is trivial; the current contour edge leads to one and only one contour edge. Case B occurs when a contour edge leads to three adjoining contour edges due to an intersection. The edge chosen is the only one of the three candidates whose orientation is consistent with the contour under reconstruction, and whose direction vector differs from that of the current edge's (Fig. 2.10). These criteria guarantee that the resulting contour will be simple, closed, and oriented.

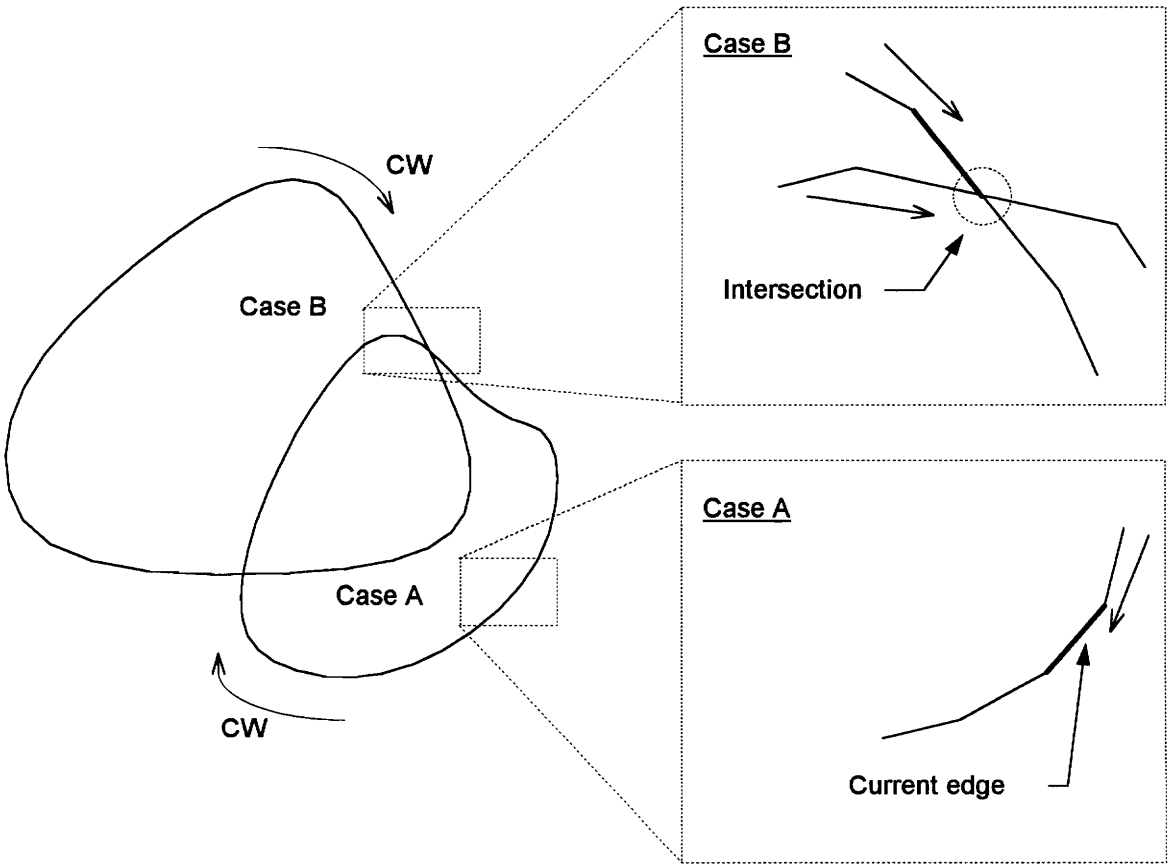
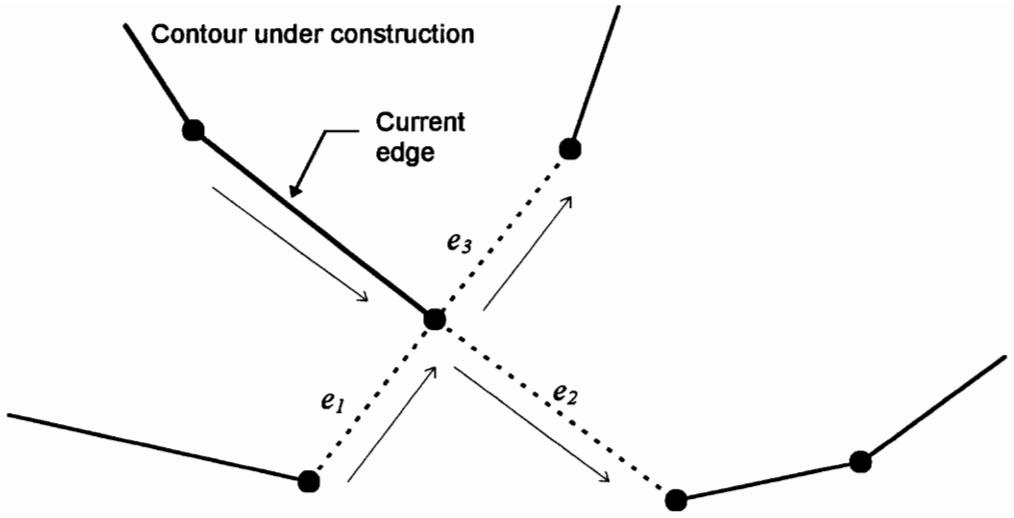


Figure 2.9: **Determining adjoining contour edges in contour reconstruction.**



e_1: Reject. Its orientation leads to the intersection point.
e_2: Reject. Its direction vector its exactly equivalent to current's (made from the same edge)
e_3: Accept.

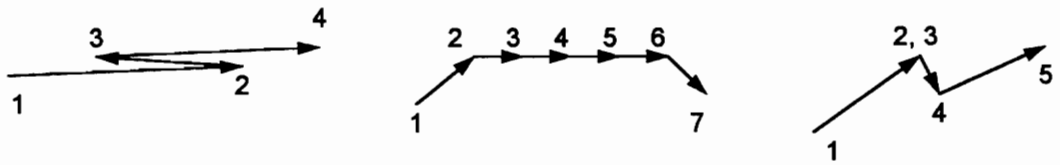
Figure 2.10: **Contour reconstruction, Case B of Fig. 2.9: the choice of the adjoining edge is based on (1) the orientation of the contour under reconstruction, and (2) on the subsequent edge direction vector.**

2.3.3 *Contour offsetting*

Contour offsetting is used both for exterior/interior separation and for contour tool path generation. For a given contour, the offset procedure evaluates the orientation of the contour and offsets it such that the resulting contour lies on the interior of the layer. This procedure requires properly closed and consistently oriented contours. The actual contour offsetting is preceded by a contour smoothing procedure in order to overcome imperfect or redundant information in the initial contour obtained by slicing.

Imperfect or redundant contour information, such as backward jerks, collinear edges, and co-incident vertices, can cause incorrect output, waste memory space and calculation time, and cause overflow problems due to non intersecting parallel edges [Yang94] (Fig 2.11). Between two consecutive edges, backward jerks and collinear edges are identified by summing their unit normals. If the resulting vector has a length approaching 2, then the normals are nearly equivalent and the edges are collinear. If the

length of the resulting vector is close to 0, then the normals are nearly opposite and the edges form a backward jerk. Co-incident vertices are identified using a distance threshold: If the distance between two vertices is smaller than a threshold limit, then the vertices are co-incident. In each case, problematic vertices, e.g., #3 in Fig. 2.11(a); # 2, 4, 5, and 6 in Fig. 2.11(b); and #2 in Fig. 2.11(c), are eliminated, and consequently, the affected edges are redefined. Each of these contour modifications save memory and simplify the contour geometry at the potential expense of losing detail.



(a) Backward jerks (b) Collinear edges (c) Co-incident vertices

Figure 2.11: **Imperfect and redundant information in slice contours.**

Once contour smoothing is performed, every edge is offset; new vertices are created, and, new edges are created from the new vertices (Fig. 2.12).

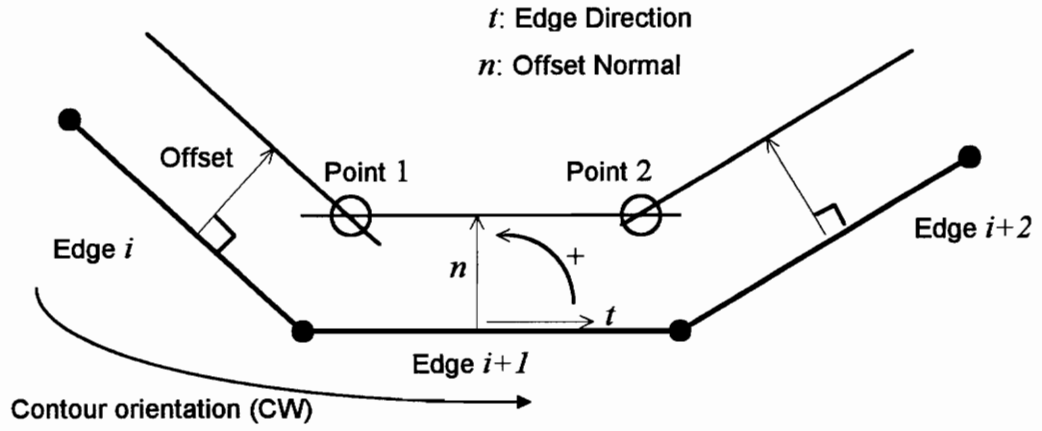


Figure 2.12: **Single contour offsetting: the edges are successively offset inward, creating new vertices and new edges. For a given edge, the offset direction is perpendicular to the directed edge vector.**

2.3.4 *Odd winding-rule*

The interior of a slice bounded by several contours can be determined by using the odd-winding rule. This rule states that a point is in the interior of the slice if a ray extending from the point to infinity intersects the slice's boundary an odd number of times (Fig.2.13). This is a classic computer graphics concept [Gaskins92], and is utilized during the contour reconstruction phase of tool path generation.

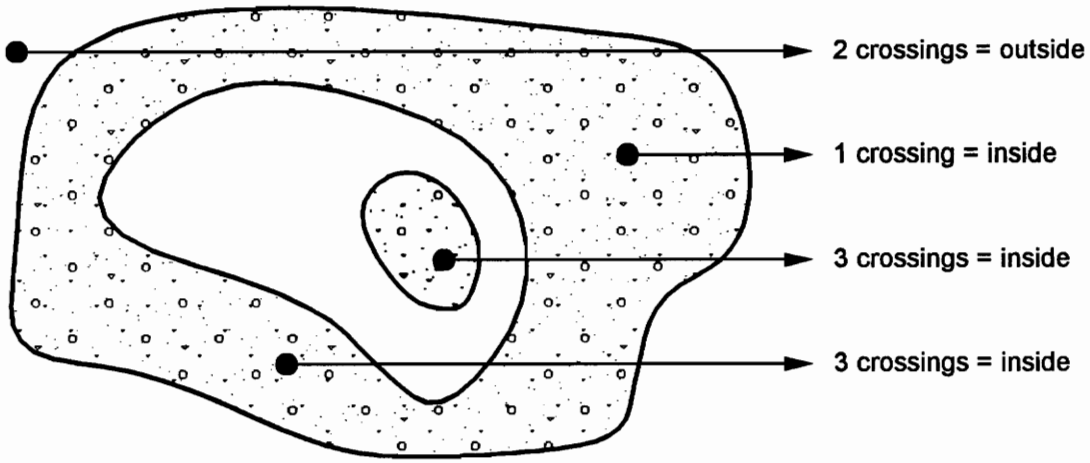


Figure 2.13: **The odd-winding rule. An odd number of crossings indicates the point is interior. An even number indicates the point is exterior.**

2.4 Applicability to rapid prototyping processes

Only some of the available commercial additive fabricators are suitable for this research. The subsequent sections group the additive fabricators according to their degree of applicability. Each section contains brief process descriptions and discusses the relevance of this work to those processes.

2.4.1 Full applicability: SLA, SLS, FDM

Stereolithography (SLA), Selective Laser Sintering (SLS), and Fused Deposition Modeling (FDM) all build a part by tracing a path over a cross-sectional layer of the part, causing the material contained in that layer to solidify and adhere to the part's previous layer [Burns93, Dickens95, Comb92, Comb94a, Comb94b]. These paths are specified using NC code, and are therefore fully customizable.

SLA is currently the most widely used rapid prototyping technology. In this process, layers of liquid photosensitive polymer are successively cured by a UV laser tracing across each layer until the final shape is obtained. SLS traces a high-power laser over a thin layer of powder in order to "sinter" the powder onto a previous layer. Although it is called sintering, the process actually melts the powder. FDM [Comb92, Comb94b] resembles the method used for building decorative icing on cakes: a robotic device extrudes a thin filament of melted material as it traces over each layer; the material bonds to the previous layer as it solidifies.

The path traced by these processes can be arbitrarily modified by changing the NC code. Furthermore, the layer thickness can be varied during fabrication. Hence, these processes can fully utilize the adaptive slicing and interior/exterior separation techniques presented later in this thesis.

2.4.2 Partial applicability: SGC, LOM

Solid Ground Curing (SGC) and Laminated Object Manufacturing (LOM) both create an entire layer at a time [Burns93, Dickens95]; therefore, the process of separating thick interior slices from refined exterior slices does not apply. However, the concept of adaptive slicing still applies.

SGC photocures an entire layer of photopolymer liquid at a time by using a photo-masking technique similar to that of circuit board manufacturing. In SGC, the variable

thickness can be achieved by changing the amount of liquid in a given layer. LOM sequentially laminates a sheet of paper to a stack of previously formed laminates, and then cuts it to the desired cross-sectional shape using a laser beam. In LOM, the variable thickness can be achieved by bonding a variable number of sheets and cutting them simultaneously.

2.4.3 Potential applicability: 3DP, BPM

3D Printing (3DP) and Ballistic Particle Manufacturing (BPM) both use a technique similar to ink jet printing [Burns93, Dickens95]. 3DP deposits a binder material onto a layer of powdered material. BPM uses a stream of particles that build up the part by impinging on a substrate [Dickens95]. The 3DP process is comparable to SLS; therefore, the proposed technique is applicable. It is not known, however, if the cross-section raster is controllable. That is, it is not known whether the solidification pattern of the layers can be modified. BPM is similar to FDM in several ways. Nonetheless, insufficient information has been published to ascertain whether the proposed work applies.

2.4.4 Other fabricators

Two techniques currently under development propose to work on dimensions much smaller than that allowed by current fabricators [Dickens95]. Laser-Induced Deposition uses a laser to "encourage" gaseous particles to adhere to a substrate. Similarly, Atomic Manipulation shows that it is possible to move individual atoms and place them at given positions on a substrate. Both processes are far beyond the scope of this research.

2.5 The .STL file format

The .STL file format is a boundary representation of 3D volume geometry using triangular facets. Each triangular facet contains the coordinates of its three vertices and a surface normal pointing away from the material. The .STL file format is limited in that it does not include interfacet topology [Rock91a], and it does not ensure intrinsically that the described solid object is closed and properly oriented [Bøhn93]. Bøhn proposes an automatic method based on the extensive use of topology to repair non-closed shells and improperly oriented solid models [Bøhn93]. This thesis uses the libraries developed by Bøhn to establish topology and to ensure that the model is closed and properly oriented.

Chapter 3

Literature Review

High fabrication speed is a major advantage of layered manufacturing technologies over traditional manufacturing. Its major limitation, however, lies in its current inability to fabricate final, functional parts. Part of this problem is due to limited material selection, and part of it is due to the difficulty of achieving an acceptable surface quality within an acceptable amount of time. Additionally, solid model preprocessing time, in particularly the slicing time, accounts for a significant part of the total fabrication time [Kirschman91]. Accordingly, this chapter reviews research on the improvement of the rapid prototyping data file format .STL, CAD model slicing principles, and the improvement of surface quality.

3.1 Model preprocessing

3D Systems, Inc. had, due to its 3-year lead on competition, the luxury of establishing what has become the rapid prototyping industry *de facto* file format standard; the .STL file format [3DSystems87]. The .STL format does not contain the specific information needed to drive the fabricators, and therefore, each additive process has a preprocessing operation that, typically, consists of slicing the .STL solid models. The subsequent sections present the issues raised by the use of the .STL file format and its impact on the slicing procedure.

3.1.1 .STL and enhanced rapid prototyping file formats

Solid freeform fabricators require specific file formats which describe the part to be fabricated. The .STL file format, the *de facto* industry standard, suffers from lack of topological information [Rock91a, Rock92, Wozny92, Bøhn93, Burns93]. Methods have

been devised to eliminate the problems [Bøhn93] and new file formats such as the RPI and the CLI formats have emerged [Rock91a, Jamieson95].

The .STL file format consists of an unsorted list of possibly unconnected triangular facets. Since the file format does not describe topology, there is no assurance that the facets combine to form valid solid models and, in particular, closed, oriented shells. Invalid solid models inhibit fabrication [Bøhn93, Bøhn95, Sheng95]. These methods all rely on the use of topology.

Topology facilitates the task of ensuring the validity of a model, for instance when checking for shell closure and proper orientation [Bøhn93]. It also facilitates many geometric operations, including slicing [Rock91b]. Finally, topology can be used to reduce the redundancy present in the .STL file format [Rock91a].

Rock and Wozny [Rock92] proposes an algorithm to extract topological information from a group of facets; in particular those in the .STL file format. Their topology reconstruction covers three sequential stages: vertex merging, face and edge creation, and face and edge relationship determination.

The .STL file format represents each facet by three vertices whose coordinates are explicitly defined. This wastes space and is a source of shell punctures. Similarly, the format also stores each facet normal explicitly as a 3-tuplet and implicitly by the order of its three vertices using the right hand rule. This also wastes space and is a source of numerical inconsistencies.

The solution to these problems is to replace redundant vertices by a single entry vertex list, and have the facets reference this list instead of sets of nine numbers corresponding to the coordinates of its three vertices. For the normal, only one of the representations needs be stored.

Rock and Wozny [Rock91a] incorporate these concepts into the .RPI file format, including storing references to adjacent facets. They meet important design considerations:

upward compatibility from .STL to .RPI, the ability to provide topological information, and CSG primitive support.

Including topology in the rapid prototyping model files would clearly speed up processing, e.g., model slicing, and ensure model validity. Lacking this information is clearly a short coming of the .STL file format. However, the simplicity of the format is one of its major strengths and is why it is so widely supported. Triangular facets represent the lowest common denominator in the field of solid modeling representations. It is therefore easy to translate to, especially since it does not require topological information. Once this information is restored, subsequent processing is trivial since it only involves linear and planar objects. The same cannot be stated for freeform modeling schemes such as non-uniform rational B-Splines (NURBS). Hence, assuming rapid prototyping models continue to be described by a set of triangular facets, the next section will examine methods to improve the ever important model slicing task.

3.1.2 Increasing slicing speed

For most layered manufacturing systems, CAD models described in the .STL file format must first be sliced into contours. This slicing process can account for 60% of the processing time [Kirschman91]. The naive approach is to intersect every facet with every slicing plane and sort to connect the resulting edges into the desired sets of contours. This is unnecessarily time consuming. Instead, the use of topology [Rock92, Bøhn93], parametric representations [Rock91b], parallel methods [Kirschman91], and enhanced geometric searching [Rock91b, Preparata88] can greatly enhance the slicing speed.

Topological information can greatly improve slicing speed. Rock and Wozny [Rock91b] demonstrated that slice contour information can be collected by marching from facet to neighboring facet. Their algorithm, described in sections 2.1.2 and 2.3.1, reduces the complexity from $O(N)+O(m^2)$ to $O(N)+O(m)$, with N being the total number of facets and m being the subset that intersects the slice plane.

The slicing process is easily parallelized, as demonstrated by Kirschman and Jara-Almonte [Kirschman91]. They make use of a simple algorithm to process .STL files on a multiprocessor machine. This algorithm naively slices the CAD model by intersecting each facet by the slicing plane, and then sorting the resulting edges. Each processor is assign a subset of the slices which then are processed in parallel. The authors report a 92% slice time reduction using 16 processors.

The slicing process is also an excellent candidate for spatial partitioning [Preparata88]. Rock and Wozny [Rock91b] improve their slicing speed by utilizing uniform spatial partitioning, with the facets being stored in specific buckets according to their location in the vertical direction. Then, instead of looking for the first facet among the whole data set, only the bucket corresponding to the height of the current slicing plane is searched. In general, uniform spatial partitioning yields significant speed improvements. Problems arise, however, if the distribution of facets is non uniform and thus the buckets contain vastly unequal number of facets. In such cases, binary spatial partitioning (BSP) [Preparata88, Naylor90] can be solution. They are, however, difficult to implement and manage, and, to date, have not been applied to slicing for layered manufacturing.

Spatial partitioning does have the potential to greatly facilitate parallelization of the above slicing algorithm. If implemented, this combination is likely to reduce the task of slicing to a computationally insignificant part of the rapid prototyping process.

3.2 Improving accuracy and surface smoothness

Research in rapid prototyping places a constant emphasis an the need for improvement of accuracy and surface smoothness. It addresses three ways to improve overall accuracy: through an exact generation of slice contours by slicing CAD original models as opposed to tessellated models [Jamieson95, Guduri92], through adapting the slice thickness in response to the surface curvature [Suh94, Dolenc94, Kulkarni95], and through the consideration of detail features that can be ignored or missed during the slicing, such as

peaks or flat areas (areas parallel to the slicing planes) [Wozny92, Suh94, Dolenc94, Kulkarni95].

3.2.1 Exact contour generation

Jamiesson and Hacker [Jamieson95] point out that a major drawback of tessellated models is that they are poorly suited for representing highly curved objects; a faceted model cannot accurately represent areas of high curvature. The authors therefore propose that the original, non-faceted model be sliced directly. Their experiments show improvements such as reduced file size, greater model accuracy, and reduced pre-processing time. Their implementation was based on a B-Rep solid model.

Guduri et al. [Guduri92] propose a similar method that generates exact contour files from a constructive solid geometry (CSG) representation. CSG solids are built upon a set of basic predefined solids such as cylinders, spheres, and boxes. These primitive solids are combined through a set of Boolean operations, such as union, intersection, and subtraction, to obtain the desired object. Guduri et al. propose to slice all primitive solids and then recombine the set of primitive contours by using the same set of Boolean operations as the entire solid is defined with. This is a fast process because primitive contours of CSG primitives are easily described mathematically.

The major disadvantage of these approaches is that B-Rep and CSG solid models are fundamentally different, and as of yet, there has still to be devised a generic method for all model formats. The above methods succeed only because they are limited to a fixed class of shapes. They do not handle general freeform shapes.

One of the motivating factors behind exact contour generation is to improve the positional accuracy. However, the positional inaccuracy within a cross section is insignificant compared to the vertical inaccuracy caused by the stair-stepping effect. This effect also degrades the part's surface appearance. At present, the only solution to improve overall surface quality is to by decrease the slice thickness. Unfortunately, this also

increases the total building time. Recent studies [Suh94, Dolenc94, Kulkarni95], however, indicate that adaptive slicing can increase surface quality without needlessly increasing overall fabrication time.

3.2.2 Adaptive slicing

Adaptive slicing attempts to increase slice density in highly convoluted regions, and reduce it wherever possible without affecting accuracy. Accordingly, a method is required for determining the optimal slice thickness. The accuracy can be controlled by a cusp height tolerance, that is, the maximum deviation allowed between the CAD or tessellated model's surface and the actual part's surface. Suh and Wozny [Suh94], Dolenc and Mäkelä [Dolenc94], and Kulkarni and Dutta [Kulkarni95] independently address this concept. Their differences lie in the computation of the layer thickness and whether physical parts have been built to confirm their theory.

For Suh and Wozny, the layer thickness is approximated using the previous layer contour. Along this previous contour, a set of sampling points is selected. The maximum allowable layer thickness is calculated at each sampling point, by considering the surface geometry at those points. Among all the computed values, the minimum is retained as the optimal layer thickness. To compute the thickness at each sample point, the part surface geometry at the sample point P is approximated by a sphere with a radius r equal to the part surface curvature ρ at that point (Fig. 3.1).

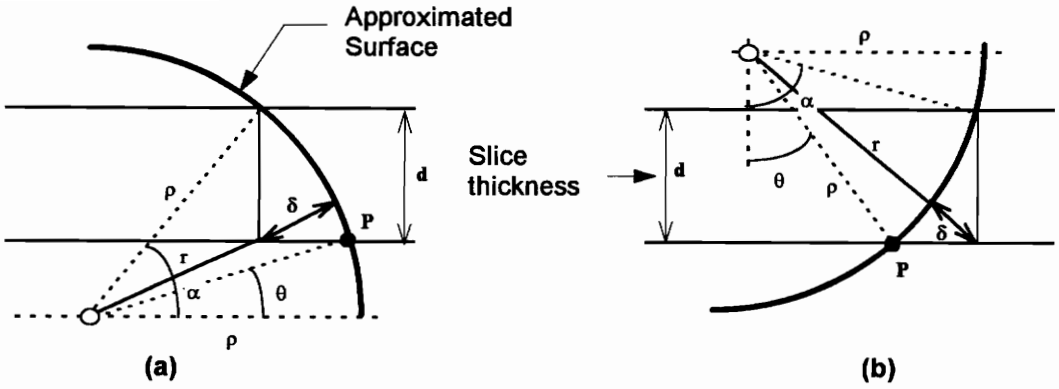


Figure 3.1: **Determining the slice thickness. The slice thickness *d* is a function of the surface curvature ρ and the user defined cusp height δ.**

The thickness d is calculated using the cusp height tolerance δ. Either equation (3.5) or (3.6) will be used, depending on to the location of the sampling point P relatively to the center of the hemisphere. In the case shown in Fig. 3.2 (a)

$$d = -\rho \sin\theta + \sqrt{\rho^2 \cos^2\theta - 2\delta\rho - \delta^2} \tag{3.5}$$

and in the case shown in Fig. 3.2 (b)

$$d = \begin{cases} \rho\cos\theta - \sqrt{\rho^2 \cos^2\theta - 2\delta\rho - \delta^2} & \text{if } \left(\rho^2 \cos^2\theta - 2\delta\rho - \delta^2\right) > 0 \\ \rho\cos\theta & \text{otherwise} \end{cases} \tag{3.6}$$

where:

d: layer thickness at the considered sampling point P;
δ: cusp height tolerance;
ρ: surface curvature at the considered sampling point P;
r: radius of the sphere; equal to the curvature ρ;
θ: angle between center of the sphere and sampling point P;

The choice of the sampling points across the horizontal layers is based on approximating the layer contours by arcs: from a given sampling point, the next sampling point on the contour is computed by approximating the contour curve by an arc (Fig. 3.2).

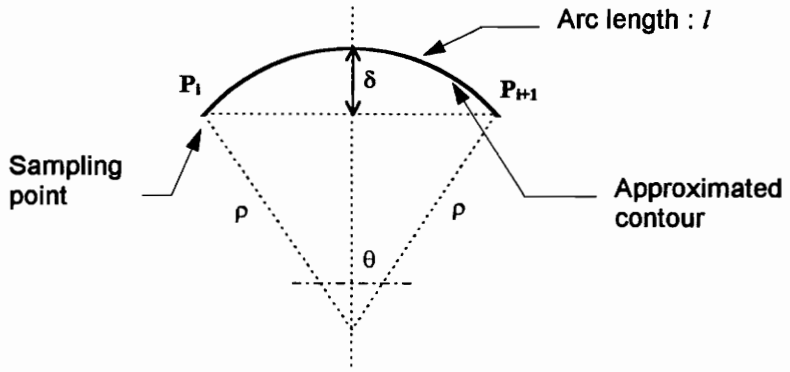


Figure 3.2: **Determining the sampling points. From a given sampling point P_i, the following sampling point P_{i+1} is a function of the radius of curvature ρ of the contour and the cusp height δ.**

$$\text{Arc length} \quad l = \begin{cases} \infty & \text{if} \quad \rho = 0 \\ 2\rho\cos^{-1}(1-\rho/\delta) & \text{otherwise} \end{cases} \tag{3.7}$$

This approach works with any CAD model for which a curvature can be found. This includes faceted models and freeform surfaces.

Dolenc and Mäkelä [Dolenc94] provide an alternative approach to adaptive slicing. Here, the layer thickness at a point P is computed based on the cusp height c, with the thickness being limited to a user-specified range $[t_{min}, t_{max}]$, and the cusp height being less than C_{max} (Fig. 3.3).

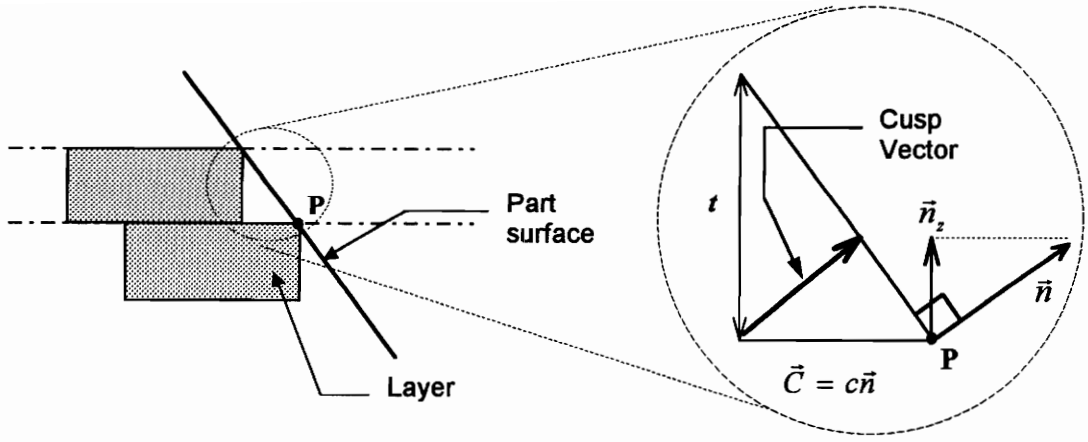


Figure 3.3: **Stairstepping effect and cusp definitions; The layer thickness t is a function of the vertical component of the normalized surface normal $\vec{n}$ and the cusp vector $\vec{C}$.**

Since the cusp vector $\vec{C} = c\vec{n}$, where $\vec{C}$ is the unit surface normal at P, must satisfy $\|\vec{C}\| = c < C_{\max}$, the slice thickness t is computed as

$$t = \begin{cases} \dfrac{\|\vec{C}\|}{n_z} & \text{if} \quad n_z \neq 0 \\ \infty & \text{otherwise} \end{cases} \tag{3.8}$$

subject to $t \in [t_{min}, t_{max}]$.

This method works for any CAD model representation for which the surface normal is known, including faceted and freeform surfaces.

Lastly, Kulkarni and Dutta [Kulkarni95] provide a method for adaptively slicing a parametrizable algebraic surface. Their approach is similar to that of Suh and Wozny [Suh94], except that they do not sample points on the surface. Instead they determine the maximal curvature analytically. Because of this analytic approach, the method is not suited for faceted models such as those described in the .STL file format

Adaptive slicing appears to be an effective approach to maximize surface quality without drastically increasing fabrication time. Suh and Wozny [Suh94] illustrate this with a 10 inch sphere: using adaptive slicing, they only required 909 layers, as opposed to 1667 layers for uniform slicing; in both cases limiting the cusp height to less than 0.006 inches. Likewise, Kulkarni and Dutta [Kulkarni95] report building an adaptively sliced ellipsoid that yielded an 18% build time improvement over a similar uniformly sliced model. In both cases the slice thickness was limited to [0.01", 0.02"] and the cusp height to 0.006". The adaptively sliced ellipsoid required 82 slices, as opposed to 146 for the uniformly sliced model. These are, to date, the only results reported.

3.2.3 Horizontal areas and peaks

Suh and Wozny [Suh94], Dolenc and Mäkelä [Dolenc94], and Kulkarni and Dutta [Kulkarni95] all emphasize that flat horizontal areas and peaks must be identified and preserved. Flat horizontal areas are areas parallel to the slicing direction, and are therefore easily missed during slicing. Similarly, peaks are small volume of material that are not detected during slicing. These problems become prominent as the slice layers grow thick. Only Kulkarni and Dutta propose a solution for handling the peaks, specifically by locating slice planes at the height of the peaks. Similarly, all the authors handle flat areas by first identifying them, and then producing slices at their corresponding heights.

In this thesis the model is first sliced into thick slices, followed by reslicing as needed. If the flat areas are not detected during the thick slicing, an error equal to that of a thick layer may result. To prevent this, the flat areas are detected before slicing by flagging all continuous groups of facets which share the same height, z, for all three vertices. The thick layers are made to match their corresponding heights.

3.3 Observations

In order to minimize processing time and maximize accuracy and surface smoothness, the literature suggests the following:

- Use spatial partitioning, topology, and marching algorithms to speed-up of the slicing procedure [Wozny92, Rock91b, Rock92].
- Use adaptive slicing to maximize accuracy and surface smoothness with minimal increase in build time [Suh94, Dolenc94, Kulkarni95].
- Detect flat areas to preserve tolerances [Suh94, Dolenc94, Kulkarni95].

Chapter 4

Methods

To achieve high speed processing with good part accuracy, the methods developed in this thesis propose to produce adaptive exterior, high speed interior layered manufacturing. The basic strategy is based on three stages: the generation of a set of thick layers, the separation of exterior and interior regions, which includes adaptive slicing, and the generation of the tool path.

The first stage is comprised of: (1) loading the model into memory, including topology organization and bucket allocation; (2) recognizing flat areas to insure optimal tolerancing; and (3) generating thick layers whose heights coincide with the flat areas, and whose thickness otherwise equals the maximum allowed by the fabricator. Also, for each layer, the maximal vertical component of the surface normal is saved for subsequent adaptive slicing. The second stage is comprised of: (1) defining the exterior and the interior regions, relative to each thick layer, by offsetting the contours into the model's interior, and by extruding these offset contours downward to the next thick layer; and (2) refining the exterior region by adaptive slicing to better approximate the original CAD model's shape. The last stage generates raster tool paths for the exterior and interior regions. Figure 4.1 illustrates the overall process.

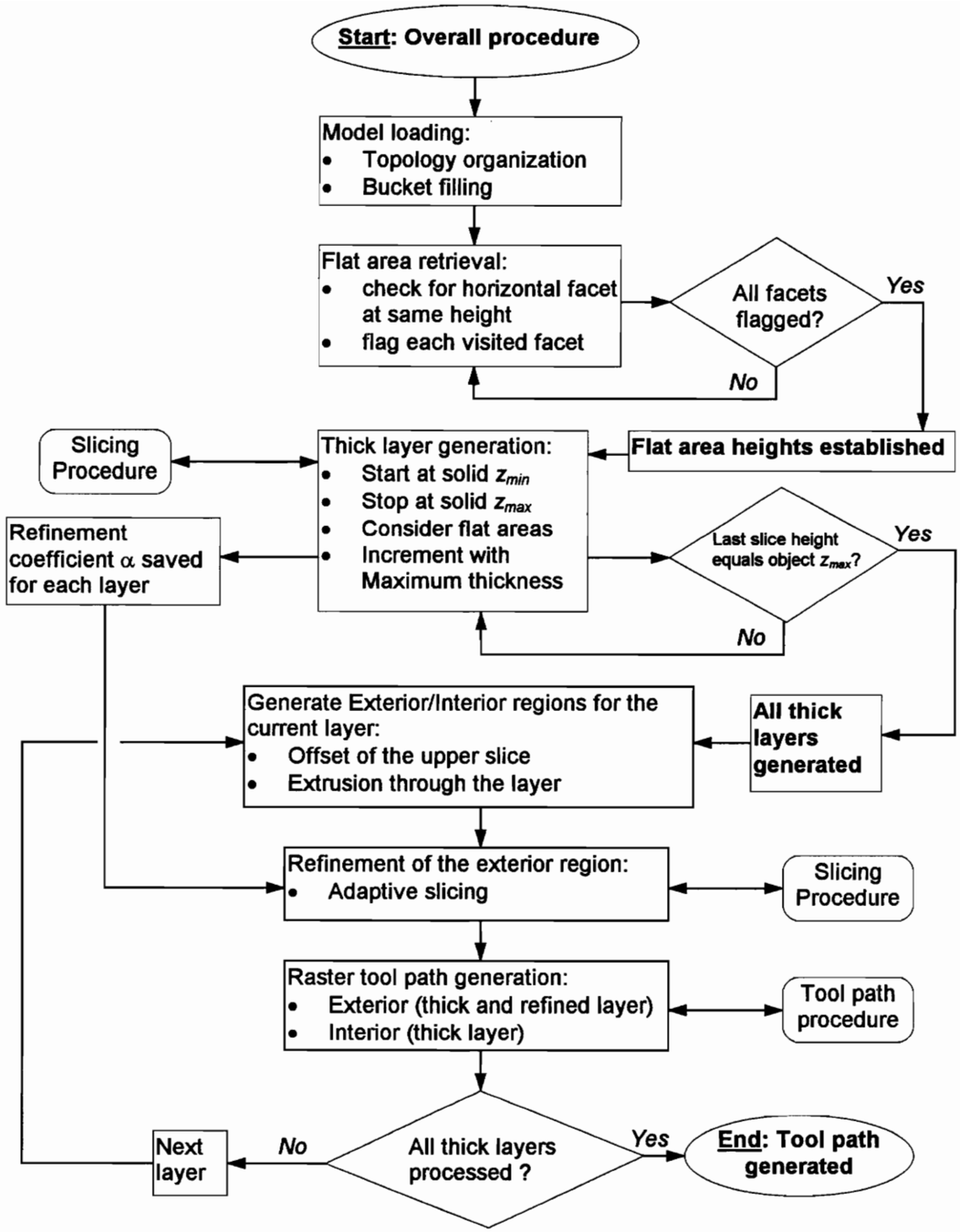


Figure 4.1: **Overall flow chart.**

Most of the methods and algorithms developed in this thesis are constrained by a compromise between speed and accuracy; when dealing with thousands of elements, speed becomes critical. On the other hand, accurate computations are needed to insure that the algorithms and the tool paths they produce, are robust. Speed and accuracy are opposing factors. To overcome this trade-off, topology and spatial partitioning have been used extensively. Topology both improves algorithm robustness and facilitates efficient searches, while spatial partitioning reduces the size of the search space in order to improve algorithm speed.

4.1 Generating thick layers

The generation of a set of thick layers initiates the process developed in this thesis. From this initial point, the exterior and interior regions can be defined, the exterior regions can be adaptively refined, and the final tool path can be generated. Four stages complete the process of creating the initial thick layers: the model is loaded into memory from an enhanced .STL file, the flat areas are retrieved, a set of slicing heights corresponding to the thick layers is generated, and, using the set of slicing height, the model is sliced using the slicing marching algorithm. Also, when performing the slicing, the maximal vertical component of the surface normal is saved for each layer. This parameter is used later to determine the amount of adaptive refinement required for each slicing height.

Thick slice generation includes: (1) specific topology organization for the facets, which includes facet/facet connectivity and facet/vertex connectivity; (2) pre-computations, which include facet normal computations, and facet minimal and maximal height; (3) uniform spatial partitioning, which reduces the model search space into uniform buckets of constant height; and (4) the actual slicing. The second stage consists of recognizing patches of horizontal facets, called flat areas, which are important to recognize in order to preserve the tolerances of the fabricated part. The third stage organizes the slicing heights under two constraints: (1) maximize the use of thick layers,

i.e., layers with a thickness equal to the maximum allowed by the fabricator; and (2) accommodate horizontal flat areas by matching the slicing heights to that of each flat area. The last stage makes use of the slicing marching algorithm to generate slices, and, consequently, the thick layers. At each slicing height, the marching algorithm: (1) finds a starting facet from the appropriate bucket; (2) uses the marching algorithm to extract the contour; and (3) repeats operations (1) and (2) until no more contours can be extracted. Fig. 4.2 illustrates the slicing procedure.

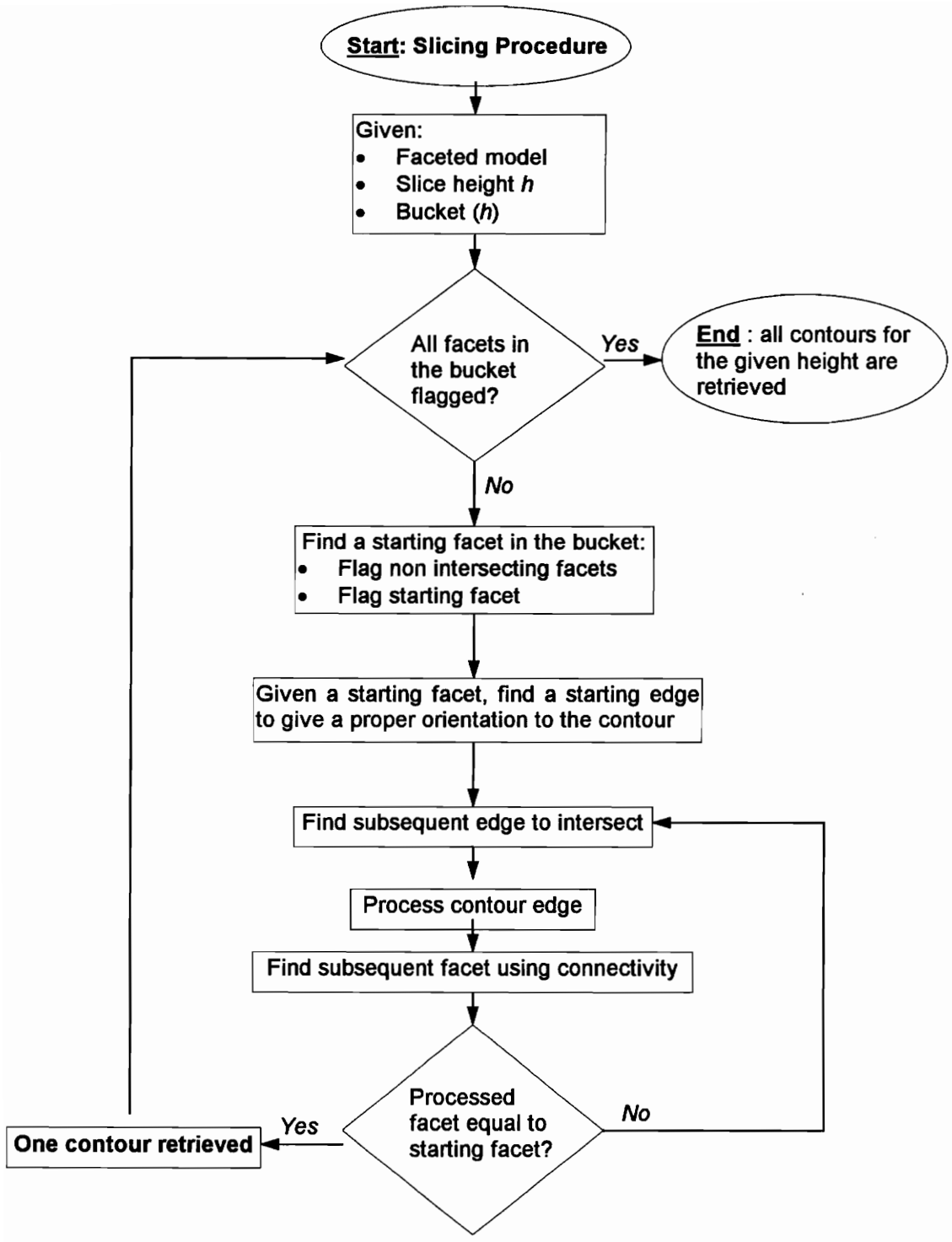


Figure 4.2: **Flow chart for the slicing procedure at a given height *h*.**

4.1.1 Model loading: topology specification, bucket allocation

Facets and vertices, as defined in section 2.1, are the two entities that described the faceted model. Its topology, which includes facet/facet connectivity and facet/vertex connectivity, is generated such that each facet contains references to its three vertices and its three neighboring facets. To give a consistent orientation to the model, each facet normal $\vec{n}$ is defined by the vertex loop sequence {*v1, v2, v3*}, following the right hand rule convention (Fig. 4.3). The enhanced .STL file provides a closed and properly oriented model, making the topology robust: no vertex is duplicated, every facet has 3 and only 3 neighboring facets, and the interior is well defined from the exterior.

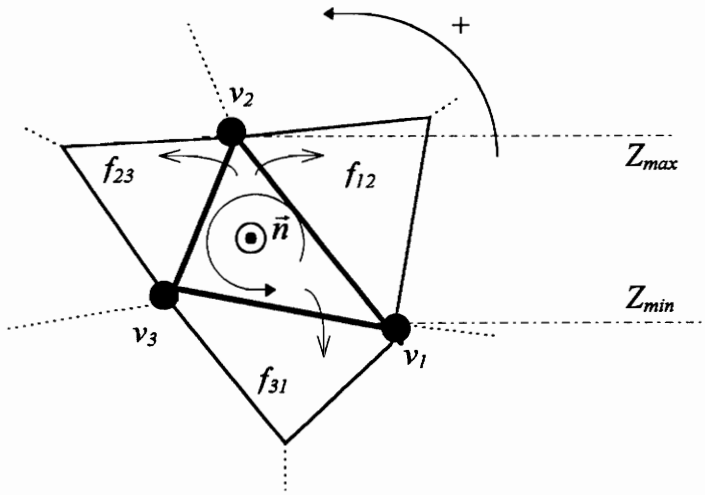


Figure 4.3: **Topological information used the slicing: each facet refers to its three vertices and its three neighboring facets, the normal is given by the vertex loop sequence. The maximal height Z_{max} and the minimal height Z_{min} are retained.**

Since the model is sliced into horizontal layers, the facets are sorted into horizontal buckets of a given height for search speed enhancement. Each bucket is defined between a z_{min} and a z_{max} (Fig. 4.4), so that, when slicing at a specific height z, the bucket taken into account is the one which includes z within its limits $[z_{min}, z_{max}]$. A facet is assigned to a bucket whenever one or more of its vertices fall within the bucket's range; if a vertex has a z value exactly equal to the boundary height between two buckets, the facet is assigned to both buckets.

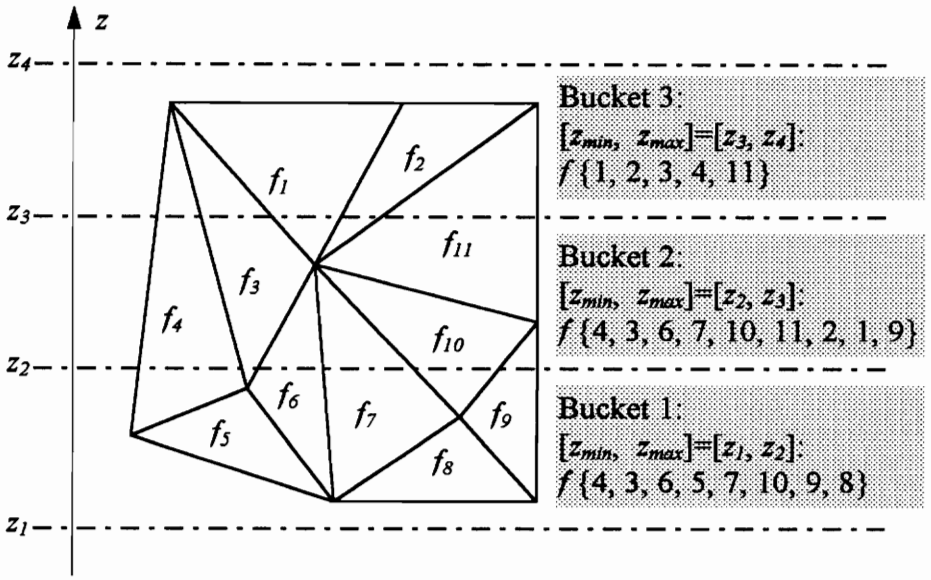
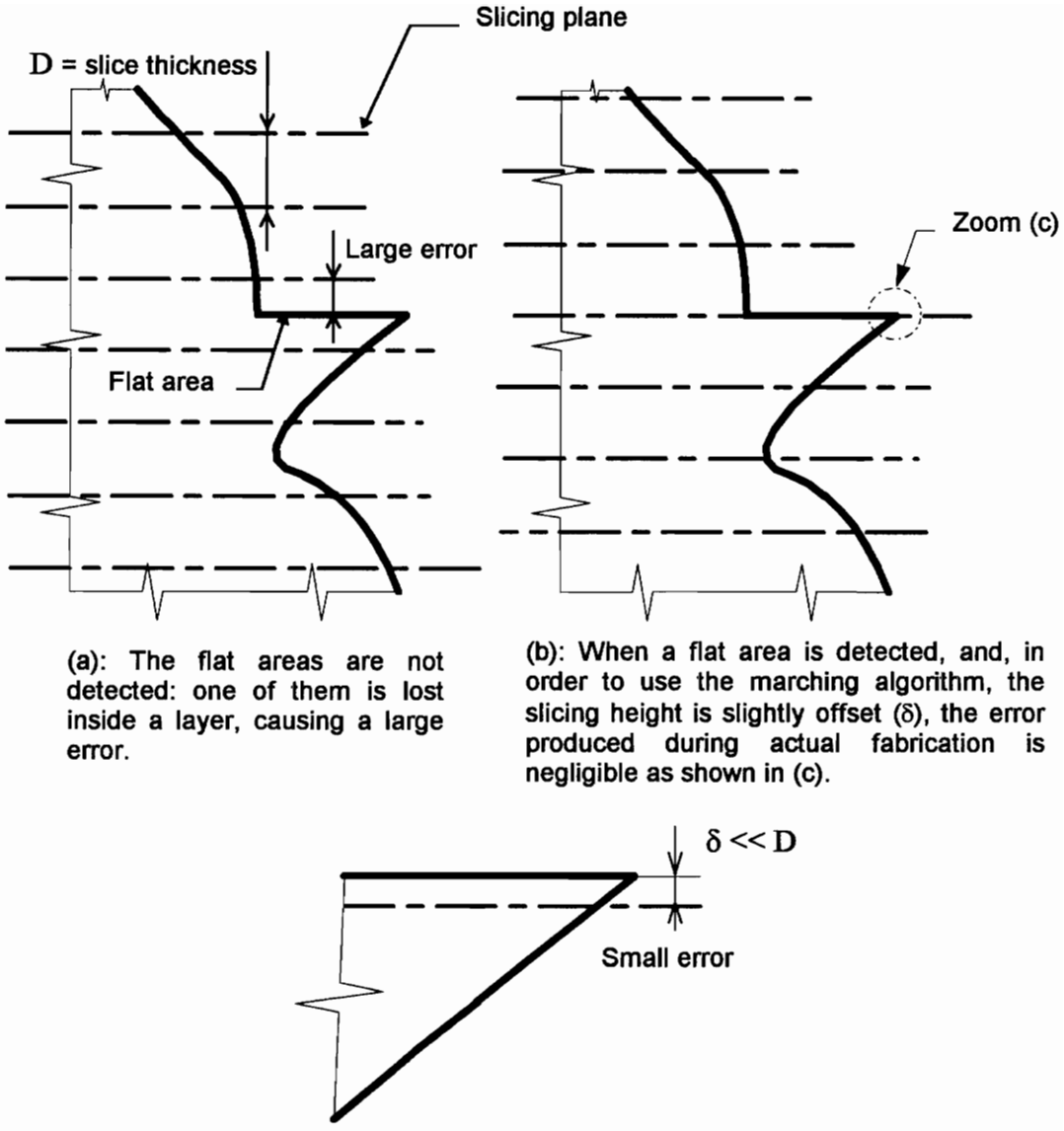


Figure 4.4: **Bucket sort: a facet which has at least one edge is partially or fully contained within a range** [zmin, zmax] **is stored the related bucket.**

4.1.2 Handling flat areas

Flat areas are patches of horizontal facets. A horizontal slicing employing uniform slice thickness cannot take them into account. This means that the difference between the height of the flat areas in a fabricated part and the height of the corresponding flat areas in the original CAD model may approach the magnitude of the sliced model's layer thickness. However, flat areas are usually critical to the overall dimensional tolerancing of the part; in some cases, the flat area may serve as a reference plane from which other dimensions are toleranced.

A horizontal facet is easily detected by comparing the z values of its three vertices: when these values are equal, the facet is horizontal. A continuous patch of horizontal facets constitute a flat area. Once a flat area is detected, the height of the slicing plane is set as an arbitrarily small offset off the flat area height (Fig. 4.5). The offset ensures that the marching algorithm will function properly, while insignificantly affecting the global accuracy of the part.



(a): The flat areas are not detected: one of them is lost inside a layer, causing a large error.

(b): When a flat area is detected, and, in order to use the marching algorithm, the slicing height is slightly offset (δ), the error produced during actual fabrication is negligible as shown in (c).

(c): The slicing height is slightly offset (δ)

Figure 4.5: **Detecting flat areas is important for tolerancing. In order to use the marching algorithm, slicing heights are computed as small offsets off the actual flat area heights.**

4.1.3 Organizing slicing heights

After the flat areas are retrieved, the slicing heights are organized under two constraints: (1) a maximum of thick layers, e.g., the thickness equals the maximum allowed by the fabricator, has to be generated; (2) the flat area heights previously determined must be accounted for. Lastly, the slicing height are sorted by increasing value.

Accordingly, from a given slice height, the next slice height is computed in two steps:

- *new slice height = current slice height + standard thickness*
- if there is no flat area between the current height and the new height, the new height is retained. Otherwise, the *new slice height = flat area height.*

Using these rules the total number of slices is minimized: a maximum of thick layers are built whenever permissible, and flat areas are built at their proper heights.

4.1.4 Model slicing

Using the set of slicing heights as defined above, the model is sliced with the slicing marching algorithm. Specifically, for each slice, the marching algorithm is repeatedly used until all the possible contours are extracted. For each contour, the marching algorithm requires as inputs a slicing height and a starting facet. Finding a contour starting facet is done by searching in the appropriate bucket for a facet whose maximal height and minimal height enclose the slicing height.

To ensure that all the possible contours have been retrieved, every facet is evaluated, either when looking for a starting facet, or during the marching algorithm. Each facet is flagged when it is evaluated. When all the facets in the bucket are flagged, no more contours can be extracted. This shows the usefulness of spatial partitioning: with no bucket, every facet in the model must be evaluated and flagged for each slice before going onto a subsequent slice.

4.2 Segregating exterior and interior regions

The segregation of exterior and interior regions is the main objective of the method developed for this thesis. From an initial set of thick layers, interior layers, and consequently, exterior shell layers can be defined. Then, each shell layer can be refined to better approximate the original model geometry. These two stages result in thick interior layers that optimize fabrication speed and thin shell layers that maximize fabrication accuracy.

The exterior and interior segregation process starts by offsetting the uppermost slice contour of each initial thick layer toward the model interior. This includes a single offset operation for each individual contour in the slice, intersection handling of the resulting set of offset contours, contour reconstruction following the intersection handling, and reconstructed contour validation (Fig. 4.6). Then the final interior contours are extruded downward through the layer to separate the interior from the exterior shell.

The second stage involves refinements of the exterior shell by adaptively slicing each thick layer into thinner layers in order to better approximate the original CAD model geometry. This last operation requires a knowledge of the original geometry. This knowledge is contained in the maximal vertical components of the surface normals of each thick layer, which were calculated and stored during the original slicing of the thick layers.

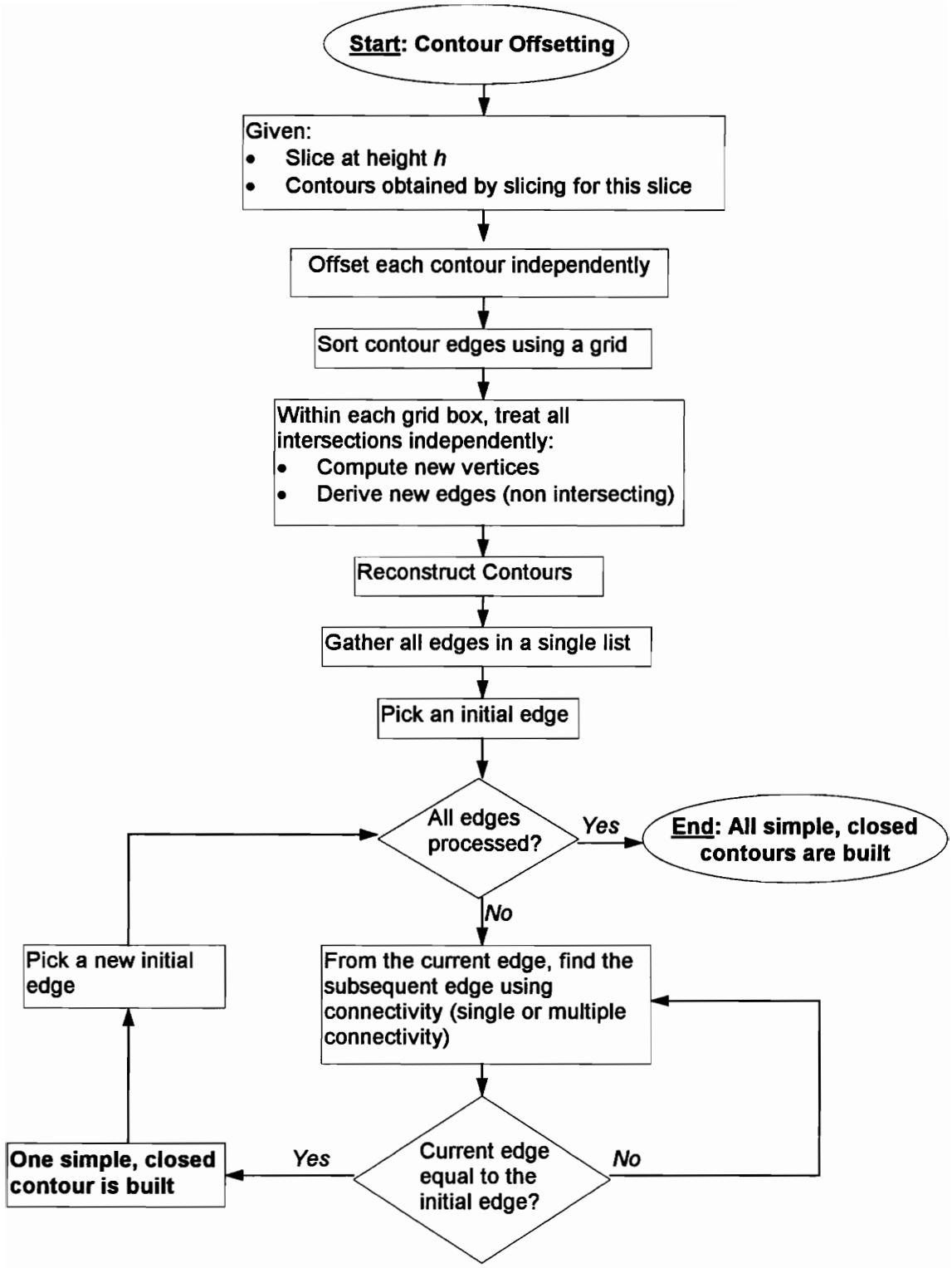


Figure 4.6: **Flow chart for the segregation of exterior/interior regions within a slice.**

4.2.1 Interior contour and shell contour definition

Within each initial thick layer, the exterior shell is defined based upon the contours of the upper slice. Specifically, the definition of the shell is based on offsetting the contours towards the interior of the model (Fig. 4.7):

- A small offset toward the interior, equivalent to half the size of the deposited material thickness, represents the outside boundary of the shell (i.e., the boundary that separates the shell from the outside, non-material region).
- A large offset toward the interior, equivalent to the desired shell thickness, represents the interior boundary of the shell (i.e., the boundary that separates the shell from the model interior). This offset must be large enough to ensure that its extrusion through the later does not puncture the model's surface; the actual value was arbitrarily set at 3 mm.

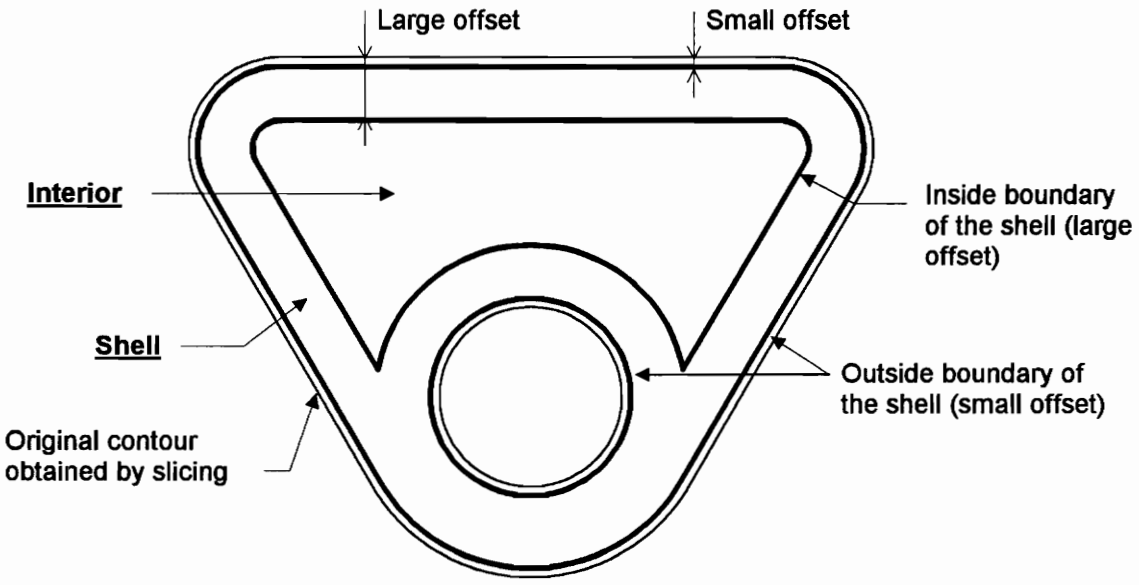


Figure 4.7: **Definition of the shell and the interior.**

Each of the above offsetting procedures are completed in sequence. The small offset operation is based upon the original slice contours. During this operation, issues discussed in section 2.3, such as recognizing collinear edges, are addressed. Resolving these issues results in offset contours which are much easier to use than the original contours in subsequent offset operations. Therefore, the large offset operations are based upon the contours produced during the small offset operation.

Offsetting each contour is a simple procedure, as discussed in section 2.3. However, two contours that do not initially intersect may intersect after offsetting. In fact, a single contour may also intersect itself after offsetting. This raises the possibility of multiple intersecting contours within a slice. In this situation, the interior of the slice cannot be defined, and therefore the tool path cannot be generated. Simple closed contours must be reconstructed from the set of intersecting contours for valid tool path generation. This can be done using a contour marching algorithm as discussed in section 2.3. This marching algorithm assumes that no edge/edge intersections exist: i.e., (1) all the intersections have been recognized, and (2) all the intersecting edges have been replaced by four distinct edges which share a common vertex (Fig. 4.8). Checking for intersections is a $O(n^2)$ operation, n being the total number of contour edges. This operation is greatly enhanced by the use of a two dimensional uniform spatial partitioning, a uniform grid, that spans the slice (Fig. 4.9). Within each of the grid boxes, the number of contour edges is much smaller than n, decreasing the cost of the computation as explained in section 2.1.

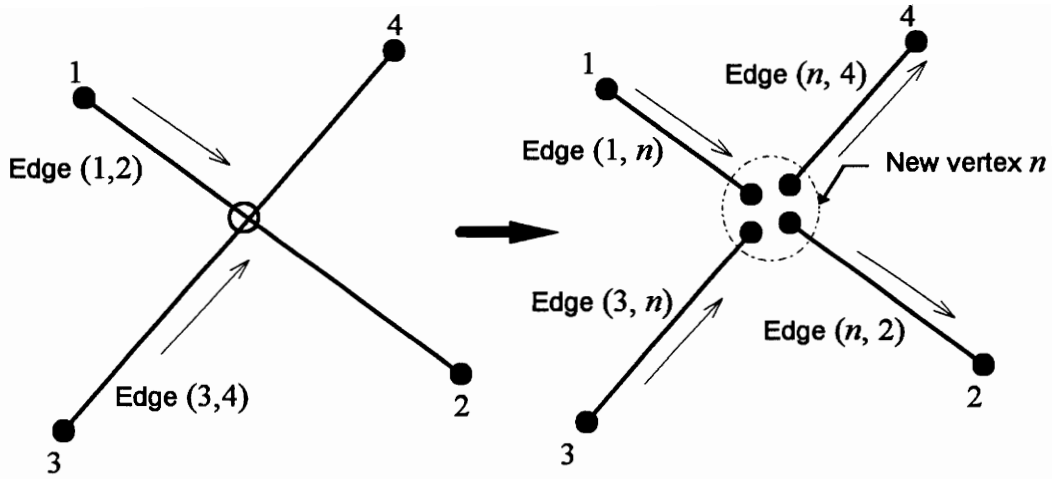


Figure 4.8: **Handling edges intersections: two intersecting edges (1,2) and (3,4) form a new vertex (*n*), and four edges: (1,*n*), (*n*,2), (3,*n*), and (*n*,4). Note that the original edge orientations remain.**

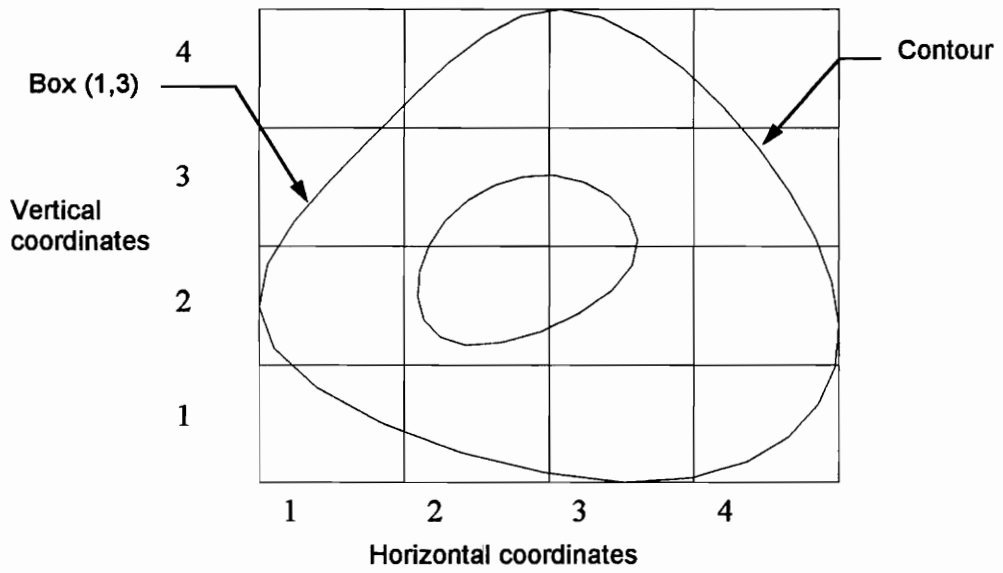


Figure 4.9: **Uniform spatial partitioning: the grid bounds the contour extremes and is aligned with the major axes (*x*, *y*).**

The marching algorithm used to reconstruct the simple closed contours guarantees that the contours have a consistent orientation. However, their orientation, i.e., either CW or CCW, remains unknown. This information is necessary to find the material boundary, and consequently, to generate a proper tool path. To find whether a contour is oriented CW or CCW, the following method is used: Given that the contour spans the 2D space (x,y), one looks for:

1. the minimum value in x
2. the minimum value in y
3. the maximum value in x
4. the maximum value in y

When traveling CCW around a contour, the above numbers are always found in a particular sequence (Fig. 4.10): 1, 2, 3, 4; 2, 3, 4, 1; 3, 4, 1, 2; or 4, 1, 2, 3, depending on the location of the starting vertex. The method applies only if all the locations of the maximum and minimum values are distinct. Otherwise, one considers the two edges that define the singular corner (Fig. 4.11). The angle formed by these edges is in the range $<-\pi/2, \pi/2>$. The sign of this angle is therefore given by the cross product of the directed vectors. If this sign is positive, the contour is CCW. Otherwise, the contour is CW.

The simple closed contours and their specific orientation defines the material boundary for the shell and the interior as explained in section 2.1: the material is inside a CCW contour and outside a CW contour. The incoherence that might appear in the material boundary, as explained in section 2.3, is eliminated through the use of the odd-winding rule.

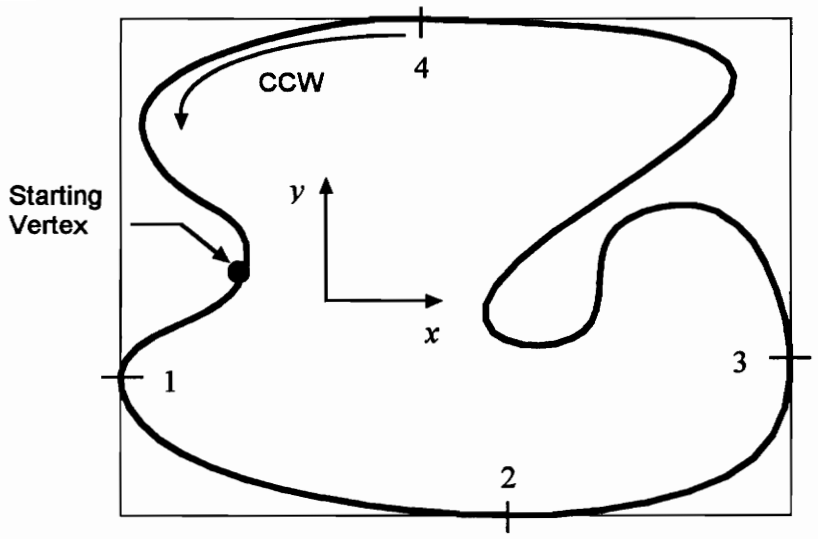
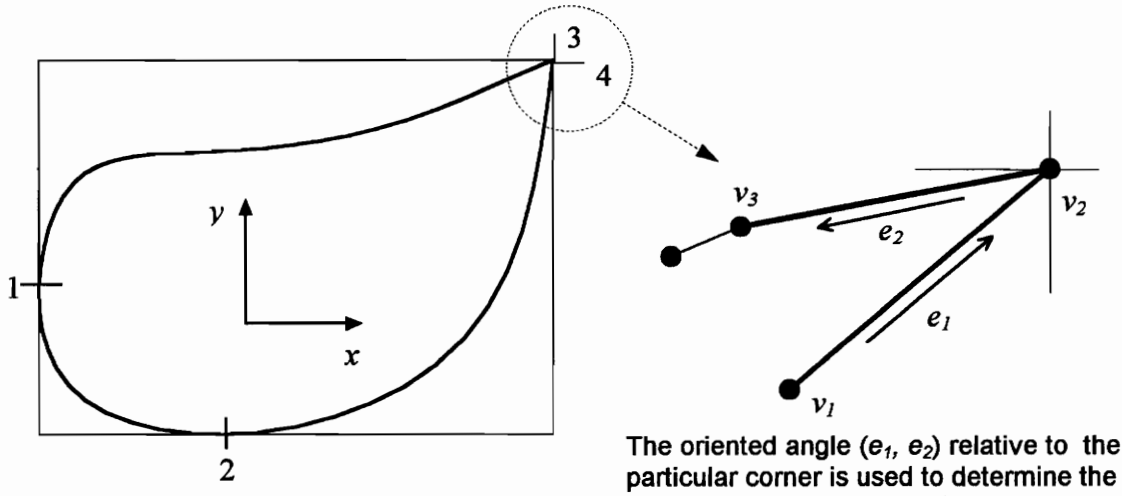


Figure 4.10: **Retrieving contour orientation: since the (x_{min}, y_{min}, x_{max}, y_{max}) sequence is 1, 2 , 3, 4, the contour must be CCW.**



The oriented angle (e_1, e_2) relative to the particular corner is used to determine the orientation of the contour: if it is positive, the contour is CCW oriented.

Figure 4.11: **Retrieving contour orientation (exception handling).**

4.2.2 Extruding the shell and interior contours

For each thick layer, the previous stage computed shell and interior contours within the layer's upper slice. For each specific layer, these contours are extruded through the layer from the upper slice to the lower slice. Extruding the upper slice's offset contours to define the interior of a thick layer, rather than forming an interpolated surface between offset contours from the upper and lower slices of the layer, greatly simplifies the segregation of exterior and interior regions. An interpolated surface would add so much additional complexity that it would drastically limit the speed of the software.

There is a remote possibility that the interior extrudes through the exterior of the shell if the extrusion is based on the upper slice of the layer alone. However, since the minimum shell thickness is about 3mm while the layer depth is about 0.5mm, the only geometry capable of producing this problem is an extreme flat overhang. Perfectly, flat areas are accounted for in determining the slicing heights, thereby limiting the problem to near-flat overhangs occurring between two thick layer slices.

With the extrusion completed, the shell and the interior are now defined in terms of layers. These shell layers are now ready for further refinements.

4.2.3 Shell layer refinement

Shell layer refinement is the final stage of segregating exterior and interior regions. At this point, shell layers and interior layers are separate entities. This procedure consists of slicing the thick shell layers into thinner shell layers to better approximate the original model geometry (Fig. 4.12). In other words, the stair stepping effect due to the layer thicknesses is minimized.

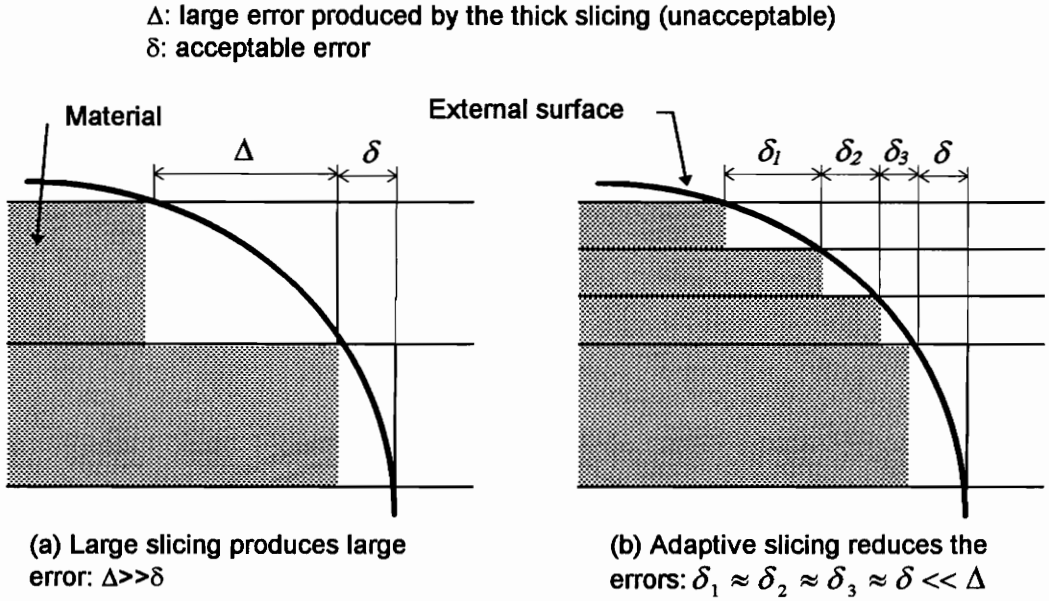
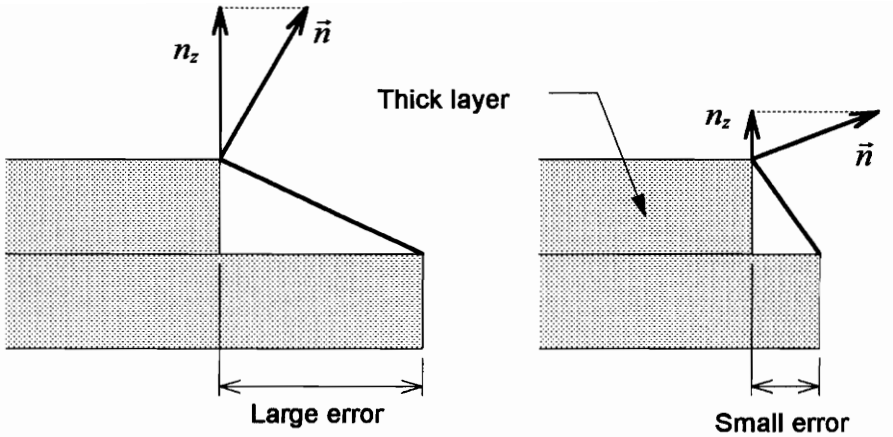


Figure 4.12: **Principle of adaptive slicing. (a): thick layers may produce unacceptable error during fabrication. (b): thinner layers reduce the errors.**

There is both a minimum and a maximum layer thickness, th_{min} and th_{max}, that the fabricator can handle. It is therefore convenient to operate with a refinement coefficient, α, an integer number indicating how many thinner shell layers correspond to a single thick layer. In essence, α indicates the shell layer refinement.

α is related to the surface geometry, and in particular the slope of the facets intersecting the contours of a layer. To facilitate evaluating the slopes, and specifically the slope leading to the worst case, the maximum vertical component of the unit surface normals of each thick slice is recorded during thick layer generation. The magnitude of this component is indicative of the refinement required in the worst case [Dolenc94], as demonstrated in chapter 3 (Fig. 4.13).



(a) Refinement is necessary, as n_z is large: the error produced between the two thick layers is significant.

(b) Refinement is unnecessary, as n_z is small. The error produced between the layers is negligible.

Figure 4.13: **The vertical component of the unit normal determines the degree of refinement needed.**

Given a cusp tolerance C_{max}, Dolenc and Mäkelä's method is adapted to match the criteria explained above. The refinement coefficient α is simply the integer part of r:

$$r = \frac{\textit{Thick layer thickness}}{\textit{Adaptive layer thickness}} = \frac{th_{\max}}{\frac{C_{\max}}{n_z}} = n_z \frac{th_{\max}}{C_{\max}} \tag{4.1}$$

subject to $\alpha \in [1, \alpha_{max}]$.

4.3 Generating the tool path

The generation of the tool path is the final stage before fabricating the part. At this point, the CAD model has been processed into a set of thick interior layers and a set of possibly thinner shell layers. The goal of this final stage is to fill these layers, using a coarse, sparse pattern for the interior and a fine, dense pattern for the shell layers (Fig. 4.14).

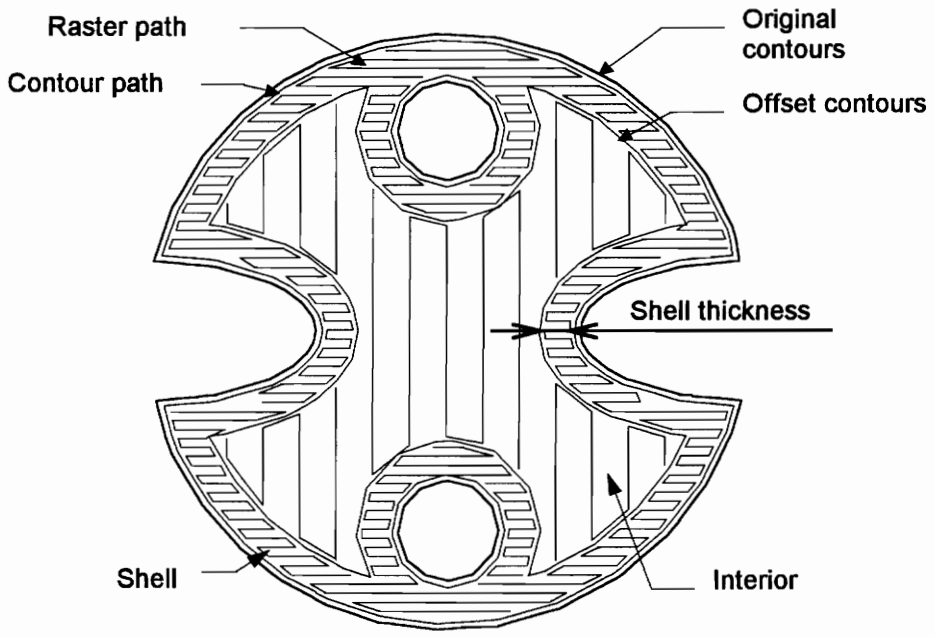


Figure 4.14: **Example of a final tool path.**

4.3.1 Defining the tool path pattern

The first step in the process of generating the final tool path is to define a general tool path pattern; there are two types of tool path patterns: contours or rasters (Fig. 4.15).

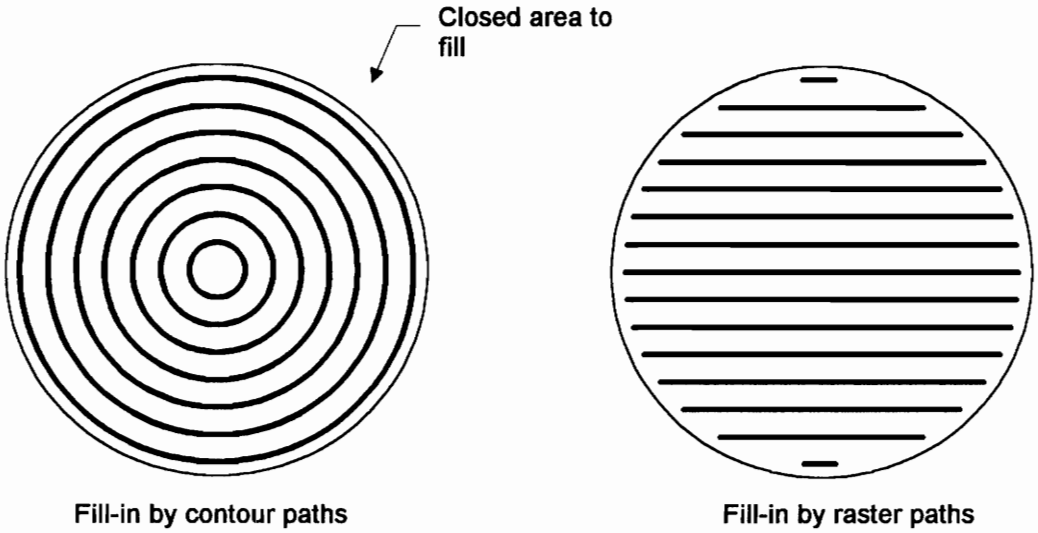


Figure 4.15 **The two ways of filling an area.**

A raster tool path is both much faster and easier to generate than a contour tool path, and, with its orientation alternating between layers, it provides stronger parts. On the other hand, contour tool path provides a smoother surface. The boundaries of the shell are therefore fabricated with contour paths to ensure a smooth external surface, while the rest is fabricated using a raster tool path pattern to facilitate a fast build.

The density of a raster pattern is controlled by the air gap between each raster segment. For the shell layers, this gap is zero to improve surface appearance. For the interior layers, the gap is typically half the thickness of the deposition. This gap facilitates faster fabrication.

4.3.2 Generating the tool path

The contour tool paths that delimit the shell have been already computed at the stage of separating the interior and exterior regions. These contours are closed and properly oriented, so no further operation is needed.

The rastering, however, needs to be computed. Fabrication processes prescribe that the rastering must be the most continuous as possible: First, it must be described by a back and forth motion to avoid any useless motion of the deposition. A back and forth motion is obtained by having the successive raster segments alternate in their direction. That is, if a given raster segment goes from left to right, the next raster segment (just above) must go from right to left. In this manner, any useless motion of the nozzle or the laser is avoided. Second, the deposition should not stop between each raster segment. Indeed, in the particular case of FDM, at each end of a deposition, the nozzle makes a z vertical motion to properly cut the bead of material. This is time consuming and, in that, should be done as rarely as possible. This can be achieved by linking the end of a raster segment to the beginning of the next one. This can be easily done, because, as previously established, two consecutive segments are oppositely oriented. Lastly, the thickness of the material deposition must be observed. The raster segments have to be slightly shortened to avoid overlapping the contour tool path (Fig. 4.14 and 4.16). Likewise, a lower threshold for the length of a tool path segment should be applied since it is, for instance, impossible to create a tool path segment with a length smaller than the thickness of the material deposition. Fig. 4.16 illustrates the shape of a raster path for a simple geometry.

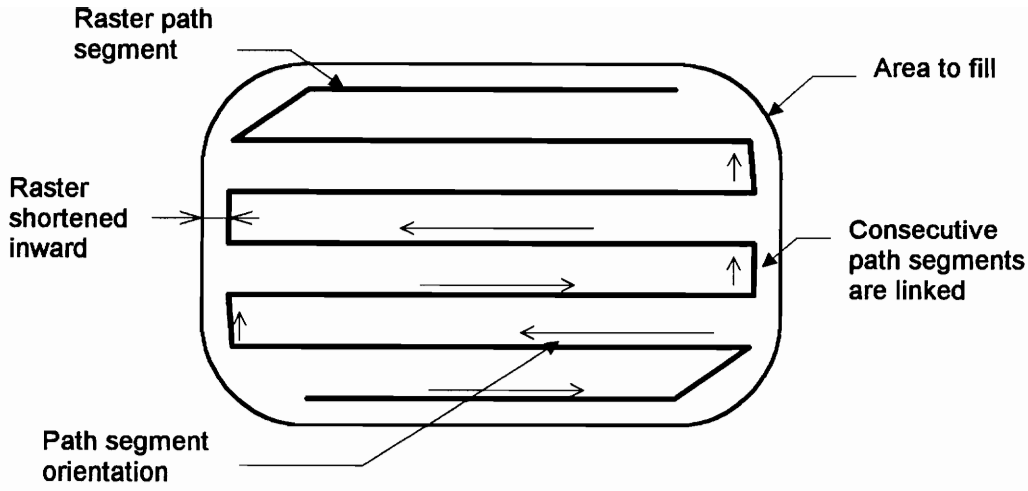


Figure 4.16: **Example of continuous motion in a raster path.**

The raster path is obtained in two stages (Fig. 4.17), subject to the constraints listed above:

- The raster path segments are generated either vertically or horizontally. Though any direction is possible, working with the major axes simplifies computations. These directions are alternated between the layers.
- The complete paths (raster segments linked together) are built from the primitive set of segments generated above.

The raster segments are generated by successively intersecting the slice contours with infinite lines as show in Fig. 4.18. The raster segments are then oriented to form a back and forth motion from a line to another.

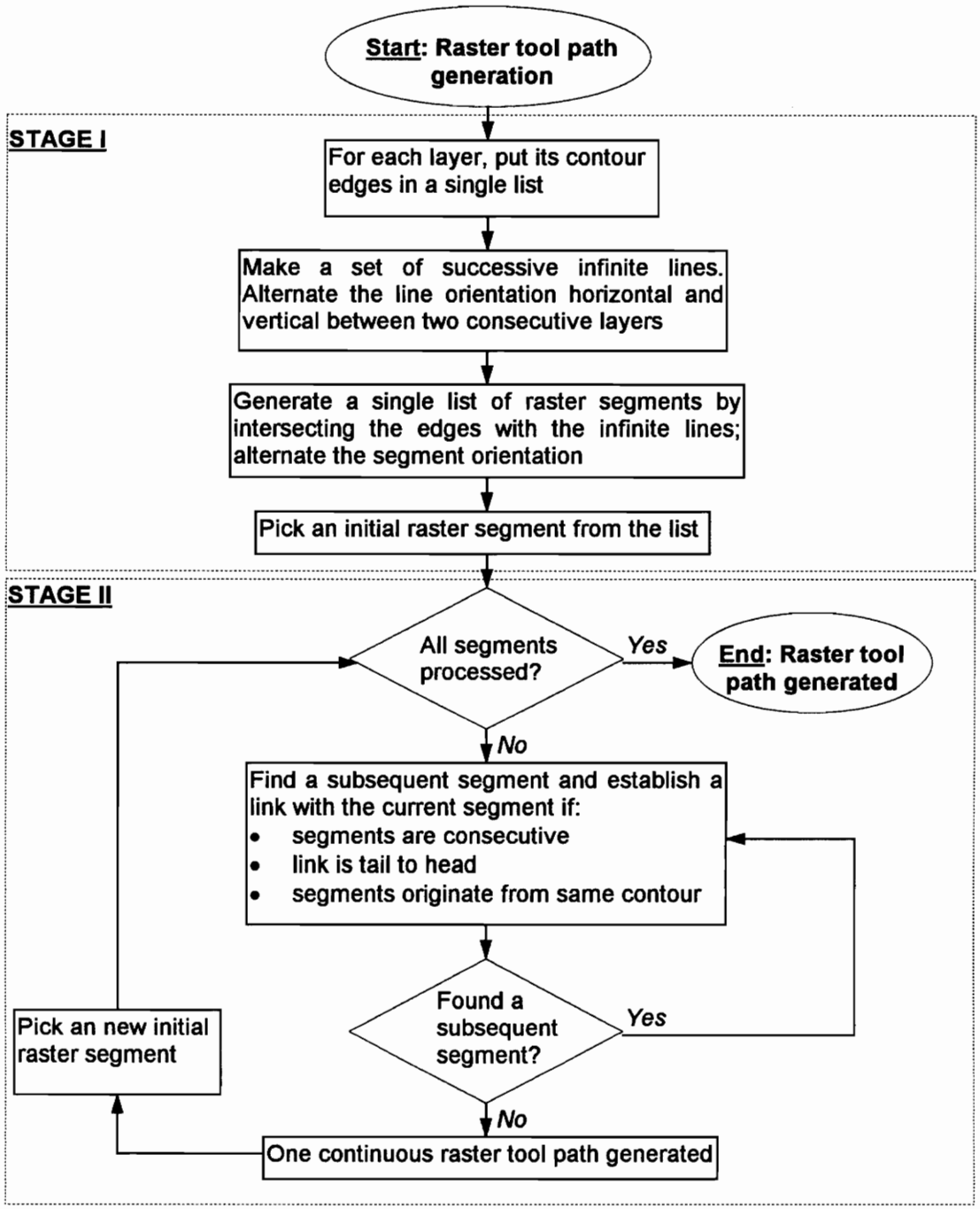


Figure 4.17: **Flow chart for raster tool path generation.**

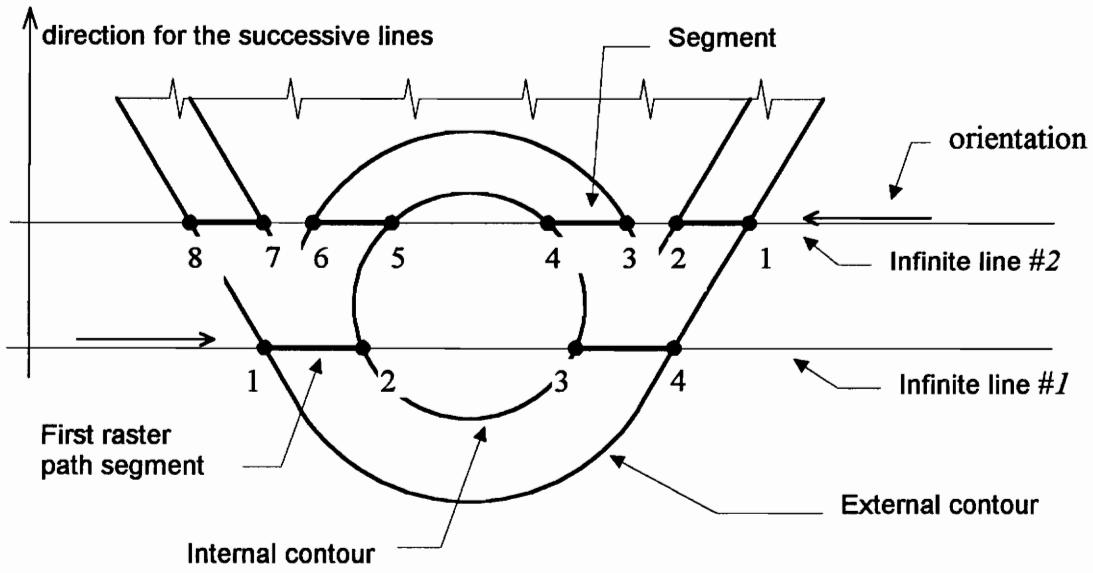


Figure 4.18: **Raster path segment creation for the shell.**

The method to make continuous chains of raster segments is based on three rules:

- A raster segment can be linked to another raster segment if and only if they are consecutive. If not, the link might cross other raster segments.
- A link is possible only between the tail and the head of a segment, in that order. This preserves proper orientation.
- A raster segment can be linked to another raster segment if and only if the two vertices to be connected originate from the same contour. If not, the link could overlap some existing raster segments.

These three rules are applied by a marching algorithm that goes from raster segments to raster segments until one of the three rules fails, which signifies the completion of a raster path. The marching algorithm reiterates until all the raster tool paths for a layer have been extracted.

Chapter 5

Results

This chapter demonstrates the feasibility of building an accurate exterior with a quickly fabricated interior. The first part of this discussion presents the slicing speeds obtained in comparison with QuickSlice, a commercial slicing program available for the FDM system. It includes an evaluation of the use of uniform spatial partitioning through two perpendicular slices of a single .STL model. The second part of this discussion presents the hardware results that have been obtained for a simple geometry. Four sample parts have been fabricated, each built with a different fabrication method: thick slicing, thin slicing, adaptive slicing, and adaptive slicing with shell/interior separation. Relative tool path length and real fabrication times are given.

5.1 Software results: slice generation speed

Two sets of experiments have been conducted to both compare the slicing speeds of the experimental code with that of commercial slicing software, and to explore the behavior of slicing with uniform spatial partitioning. The benchmark software is QuickSlice 1.4a from Stratasys, Inc.

5.1.1 Experiments

The first set of experiments utilized a model containing 8948 facets representing a human's nasal cavities. The model was constructed from 27 computer tomography (CT) scan cross sections obtained from medical imaging. It therefore contained a large number of facets which were highly stretched in the vertical direction due to the reconstruction from the 27 dense cross sections; none of these facets, however, spanned more than $1/26^{th}$ of the

vertical space. The model is elaborate enough for meaningful measurement of slicing speeds. The second set of experiments used the same model; however, this time the part was rotated by 90 degrees about the x axis in order to better randomize the vertical distribution of facets. The uniform slice thickness was set to 0.01 mm, resulting in 660 slices for the original model (Fig 5.2) and 775 for the rotated model. In both sets of experiments, the vertical resolution of uniform spatial partitioning varied, with the number of buckets going from 1, which is equivalent to no partitioning, to 100. Statistical data on facet distribution inside the buckets was collected for each spatial partitioning configuration.

The slicing was been performed on an SGI Indigo2 with 128 Mb of RAM. The C++ program was compiled with the GNU C++ compiler using the -O3 optimization option. The results are shown in Table 5.1. and Fig. 5.1.

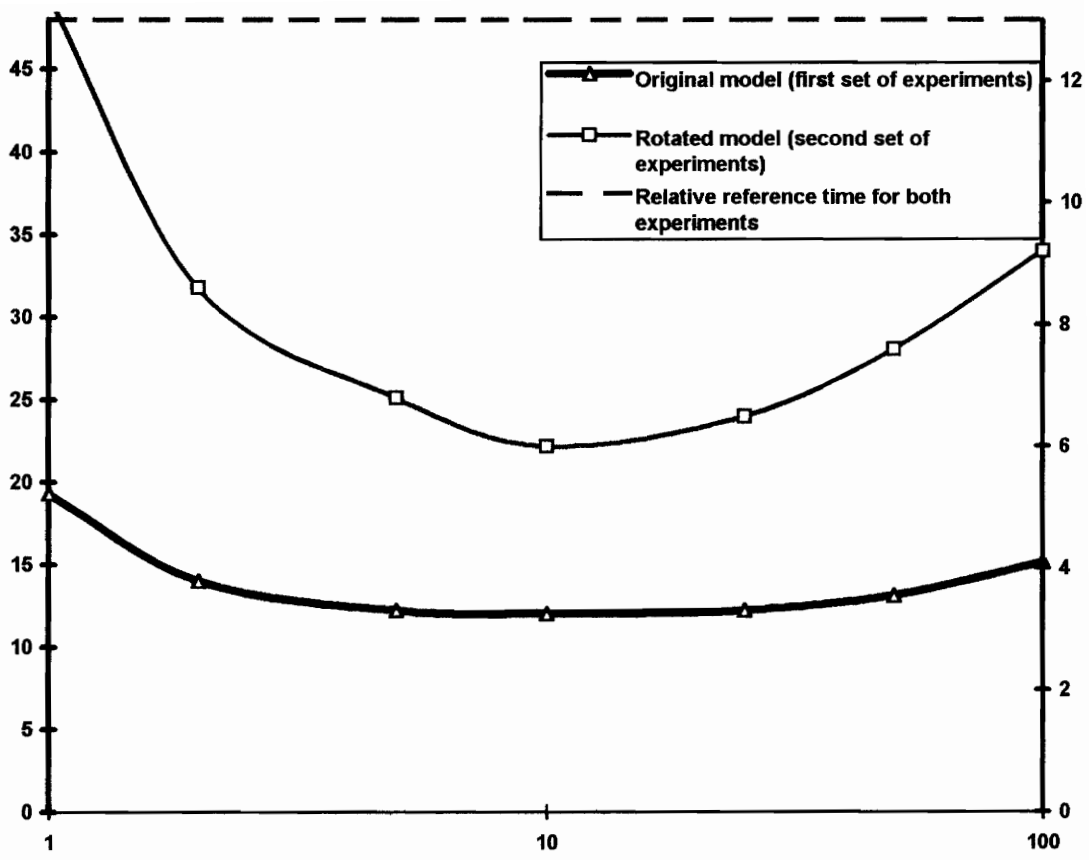


Figure 5.1: **Comparing relative slicing speeds with varying the number of buckets. The left Y axis measures the time taken for slicing the original model. The right Y axis measures the time taken for slicing the rotated model. The Y axises are scaled so that the slicing times taken by QuickSlice can be represented by a flat, horizontal line.**

Table 5.1: **Slicing speed results**

	Original model					
	Facet/bucket distributions					Slicing Time (s)
# of Buckets	Total	Avg.	Max.	Min.	Std. Dev.	
1	8,948	N/A	N/A	N/A	N/A	**19.3**
2	9,762	4,881	6,163	3,599	1,813.2	14.0
5	12,035	2,407	3,410	1,499	782.3	12.2
10	15,810	1,581	2,347	927	500.0	**12.0**
25	26,950	1,078	1,956	383	507.3	12.2
50	45,600	912	1,956	383	425.4	13.1
100	82,900	829	1,956	383	350.7	15.1
QuickSlice (reference time)						**48**

	Rotated model					
	Facet/bucket distributions					Slicing Time (s)
# of Buckets	Total	Avg.	Max.	Min.	Std. Dev.	
1	8,948	N/A	N/A	N/A	N/A	**13.4**
2	9,250	4,625	4,686	4,564	86.3	8.6
5	10,110	2,022	4,404	418	1,615.2	8.0
10	11,210	1,121	2,421	170	845.2	**6.0**
25	14,775	591	1,622	95	417.7	6.5
50	20,600	412	1,144	95	271.4	7.6
100	32,400	324	872	95	195.0	9.2
QuickSlice (reference time)						**13**

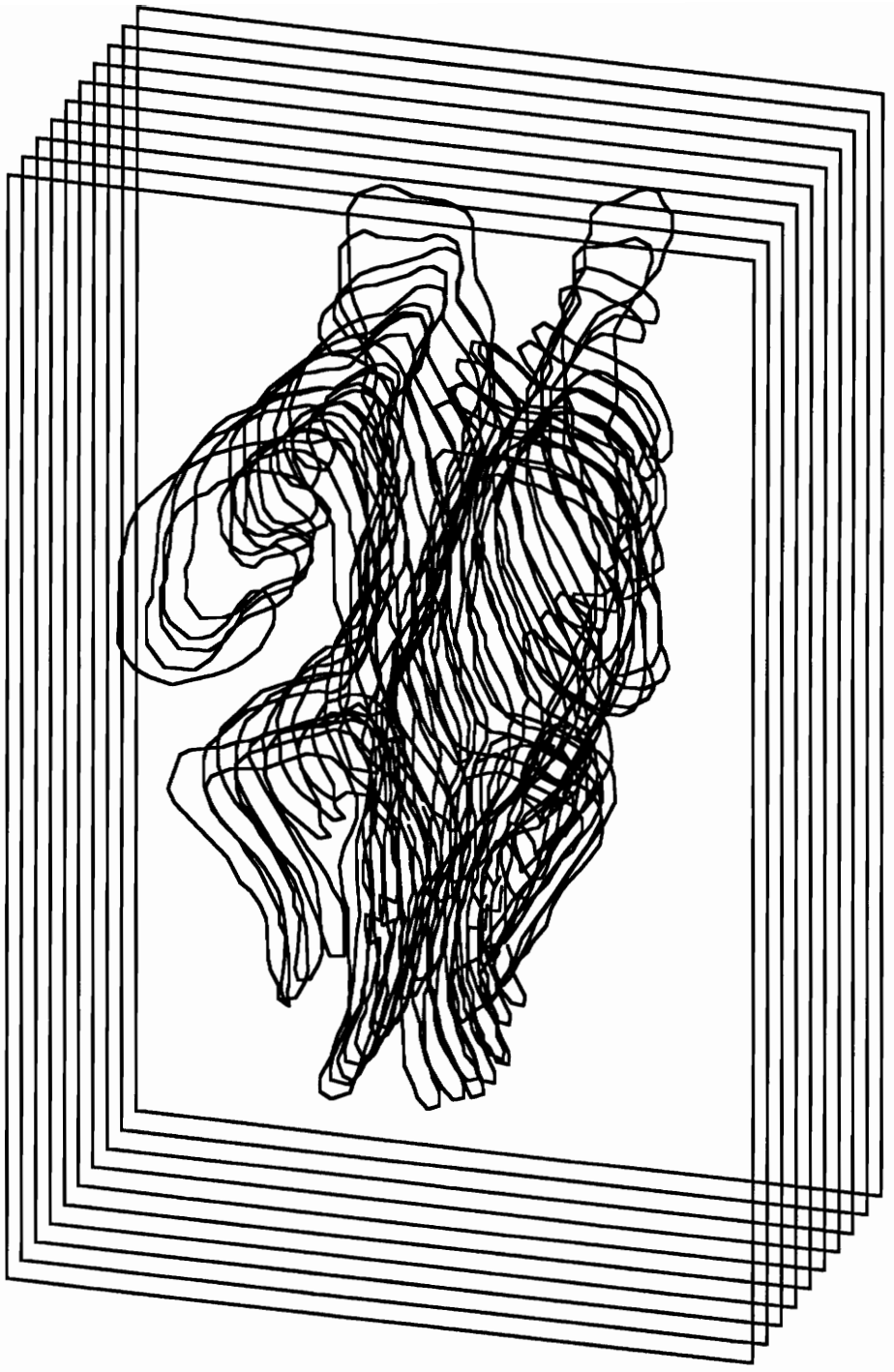


Figure 5.2: **Original model representing nasal cavities (reduced number of slices).**

5.1.2 Observations

The following observations apply to both models:

- Some facets are members of more than one bucket, even when using only two buckets.
- The number of multiple-bucket memberships increases with the number of buckets: 82,900 instances for 100 buckets with original model; 32,400 instances for 100 buckets with the rotated model.
- The slicing time reaches a minimum (12 seconds for the original model, and 6 seconds for the oriented model) for 10 buckets.

These three points suggest the following: (1) There is an optimal number of buckets beyond which further partitioning does not help. (2) The optimal number of buckets seems to be 10 for this particular data. However, a conclusion upon the actual optimal number of buckets cannot be based on two models alone. Further experiments are therefore needed to better understand the optimal number of buckets for each model to be fabricated. (3) The experimental software slices the solid model faster than QuickSlice. For the two experiments, the timing was 12 vs. 48 seconds and 6 vs. 13 seconds, respectively (Fig. 5.1).

The experiments confirm that increasing the number of buckets beyond a certain point only marginally decreases the search spaces. From this observation, coupled with the fact that increasing the number of buckets also increases the amount of time spent organizing the buckets, it becomes clear why the program reaches an optimal number of buckets, and why its overall speed decreases when this number is exceeded.

It was also observed that without the use of buckets, the original model sliced in 19.3 seconds, as opposed to 48 seconds with QuickSlice. For the rotated model the times dropped to 13.4 and 13 seconds, respectively, which suggests that both the experimental

program and QuickSlice prefer a slicing orientation in which less facets are intersected at a given slicing plane: In the first experiment, the facets were highly stretched in the vertical direction, yet densely grouped along the horizontal plane. Hence, each resulting contour contained a high number of edges. When the model was rotated, the facets became highly stretched in the horizontal direction and the contours became less complex since less facets were intersected per slice. Even tough high aspect ratio facets tends to produce more contour edges, since each facet will tend to be intersected by a larger number of slice planes than facets with a low aspect ratio, this is not the main factor in determining slicing speed. The key parameter is the total number of intersecting facets per slices.

5.2 Hardware results

The concept of adaptive high-precision exterior, high-speed interior layered manufacturing was demonstrated with the building of four parts on an FDM 1600 rapid prototyping machine. The parts were of the same original .STL model but built using four different methods: (1) building with uniform layers of thickness equal to the maximum recommended by the fabricator; (2) building with uniform layers of thickness equal to the minimum recommended; (3) building with adaptive slicing; and (4) building with adaptive high-precision exterior, high-speed interior (Fig. 5.3). Each tool path used a dense raster pattern with a width of 0.020" (0.5 mm). In addition, for the adaptive high-precision exterior, high-speed interior part, the interior was sparsely built with a 0.010" (0.25 mm) air gap between rasters. All parts kept proper functional shape and acceptable durability.

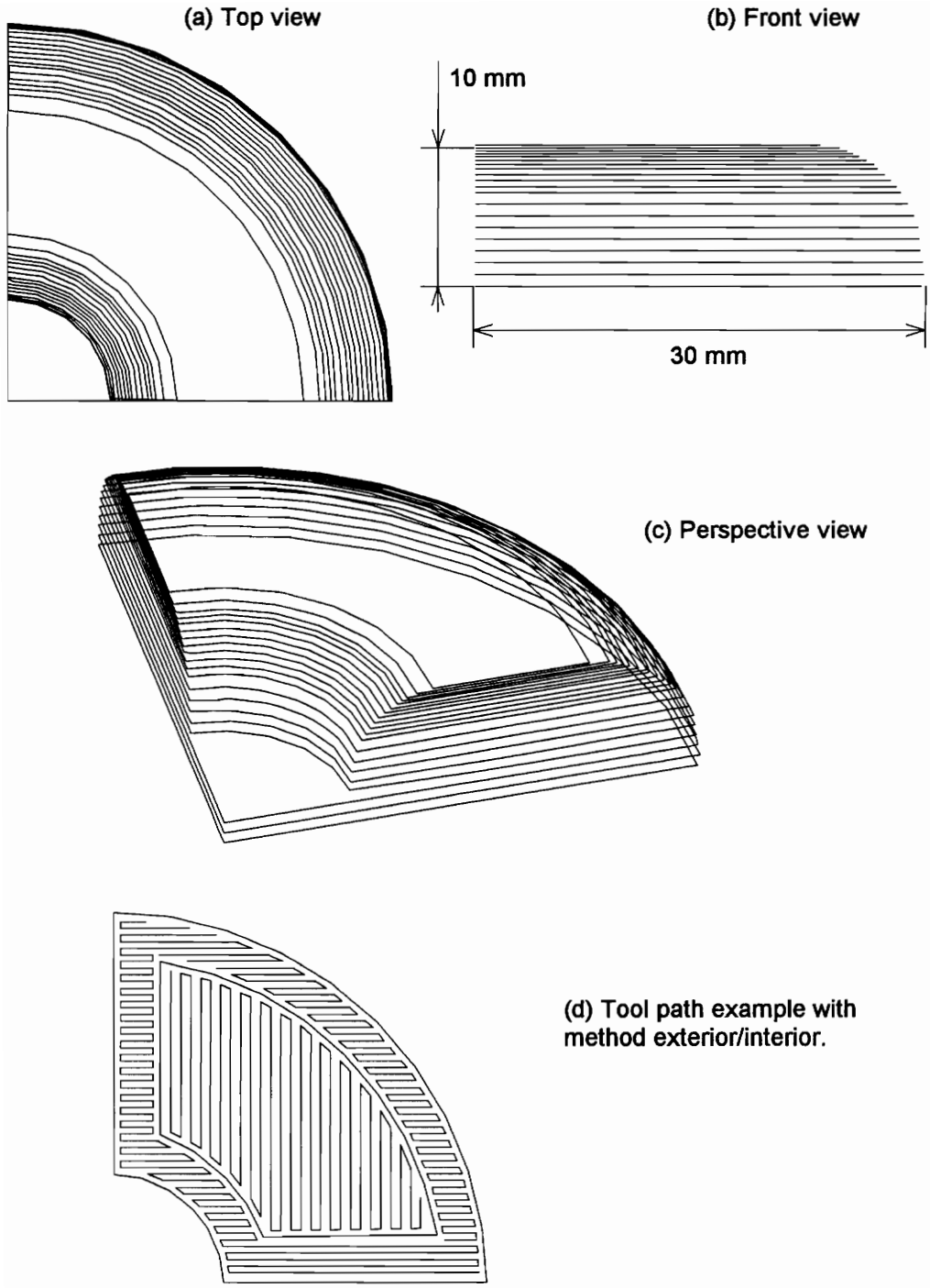


Figure 5.3: **Test part built (reduced number of slices).**

A comparison of accuracy between the four methods cannot take place until the concept of accuracy is defined and understood. Three types of accuracy can be considered: (1) The total accuracy, which is a measure of how well the dimensions of the fabricated part match those of the original CAD model; (2) the slicing accuracy, which is a measure of how well the sliced model dimensions match those of the original CAD model; and (3) the fabrication accuracy, which is a measure of how well the fabricated part's dimensions match those of the sliced model. Since the fabrication accuracy is both machine and material dependent, this thesis does not address this factor. Since total accuracy is affected by fabrication accuracy, this factor cannot be addressed either. Therefore, the focus of this discussion will be on the slicing accuracy.

The slicing process is purely computational, as opposed to physical, which means that its accuracy must be measured by evaluating a suitable error parameter. The parameter used both in this work and in the literature [Dolenc94] is the cusp height between the layered model and the original model. The adaptive slicing criteria used guarantees that the maximum cusp height is equal to that of the 0.005" sliced model. Therefore, the adaptively sliced model has the same theoretical accuracy as the 0.005" model.

The number of slices, the relative tool path length, and the actual building times for each method is listed in Table 5.2.

Table 5.2: **Hardware results**

Method	Layer thickness	No. slices	Tool path length (mm)	Relative tool path length	Actual build time (h)	Relative build time	Matl. use (g) +/- 0.001	Relative Matl. use
(1) Uniform thick layers	0.0150"	25	9,843	40	0.5	36	5.558	100
(2) Uniform thin layers	0.0050"	75	24,710	100	1.4	100	5.312	96
(3) Adaptive slicing	0.0050" 0.0075" 0.0150"	38	17,885	72	0.85	61	5.295	95
(4) Shell/Interior	0.0050" 0.0075" 0.0150"	38	13,667	55	0.81	59	4.536	59

The adaptively sliced model required 0.85 hours to fabricate as opposed to 1.4 hours for the thin sliced model. This represents a gain of 39% in build time. The part built with the method developed in this thesis was built in 0.81 hours. This represents a gain of 42% with respect to the thin sliced model. Although the latter is faster, the difference in building time is minor. The main reason for the small difference is that the tool path for the thesis method is intrinsically more complex, and, since it has not been optimized, contains excessive wasted motion. The larger difference in relative tool path length, 72 vs. 55, shows the potential of the thesis method. If fully optimized, it should be able to fabricate the part in 0.65 hours. This is 30% slower than the fastest speed attainable by using 0.015" layers, but 115% faster than with uniform high-precision thin layers and 31% faster than with regular high-precision adaptive layer thicknesses.

The surface quality of the adaptive slicing and the new shell/interior method appears, as expected, to be equivalent to that of the thin uniform slices, and noticably smoother than the thick uniform slices.

Chapter 6

Conclusion and Recommendations

This thesis has presented a method for layered manufacturing technologies to build parts described by faceted models both accurately and fast. It has also presented actual fabricated parts which demonstrate the feasibility of the method. This work answers a need in rapid prototyping research for a simultaneous improvement in accuracy and processing speed, and represents another important step toward a fully automated, accurate, and fast layered manufacturing process. This chapter concludes this thesis with a series of final remarks, contributions, and suggestions for future work.

6.1 Concluding remarks

Slicing algorithms have been implemented with a constant attention to speed. The slicing speed has proven to be faster than current commercial systems. The slicing process can be improved further by automatically optimizing the spatial partitions, for instance by employing BSP.

Although the tool path generation algorithm employs uniform spatial partitioning, it needs to be optimized with respect to tool motion and support generation. The current code was only developed to demonstrate concurrent thin and thick layers.

6.2 Contributions

This thesis addresses the dilemma of choosing between part accuracy and process speed, a dilemma currently faced by the rapid prototyping industry. Specifically, this thesis has made the following contributions:

High speed slicing: the importance of topology and spatial partitioning in high speed slicing has been confirmed.

High part accuracy with low building time: the viability and benefit of concurrent high-precision exteriors and high-speed interior region has been demonstrated.

6.3 Recommendations for future work

This thesis represents an important step toward a faster and more accurate rapid prototyping technology. However, several research directions should be investigated.

Improved spatial partitioning: The use of spatial partitioning greatly enhances the slice generation speed. However, optimal spatial partitioning is model dependent. Hence, the spatial partitioning needs to be improved to further speed up the slicing process.

Improved process parameters: The thickness of the exterior region has been set to 3 mm. This appears to be a safe value that avoids the problem of the interior puncturing the shell's exterior surface, and it ensures a feasible raster tool path. More experiments, however, are needed to determine how this thickness should be set.

Tool path optimization: Current tool paths for layered manufacturing are based on the use of linear tool path segments. The controllers of most additive fabricators, however, possess the capability of handling arced tool paths. The use of such arcs in addition to linear segments would greatly compact the memory requirements for tool path data, especially for freeform shapes.

Adaptive tool path width: Current state of the art tool path generators deposit material in constant widths. However, the possibility exists to employ continuously variable tool path widths. Specifically, each tool path segment, whether an arc or a line, would linearly vary in width between its beginning and its end. In this manner, problematic features such as corners could be filled more densely to improve part accuracy and integrity.

References

[Bøhn93] Bøhn, J.H., *Automatic CAD model repair*, Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY, August 1993.

[Bøhn95] Bøhn, J.H., "Removing zero-volume parts from CAD models for layered manufacturing," *IEEE Computer Graphics & Application*, Vol. 15, No. 6, November 1995, pp. 27-34.

[Burns93] Burns, M., *Automated fabrication; improving productivity in manufacturing*, PTR Prentice Hall, Englewood Cliffs, NJ, 1993.

[Comb92] Comb, J.W., Priedeman, W.R., and Turley, P.W., "Rapid Prototyping using FDM: a fast, precise and safe technology," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 3-5, 1992, pp. 301-308.

[Comb94a] Comb, J.W., Priedeman, W.R., and Turley, P.W., "FDM technology process improvements," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 8-10, 1994, pp. 42-49.

[Comb94b] Comb J.W., Priedeman W.R., and Turley P.W., "Layered manufacturing control parameters and material selection criteria," *Manufacturing Science and Engineering*, Vol. 68-2, ASME 1994, pp. 547-556.

[Dickens95] Dickens, P.M., "Research developments in rapid prototyping," *Journal of Engineering Manufacture*, Vol. 209, Part B, January 1995, pp. 261-266.

[Dolenc94] Dolenc, A., and Mäkelä, I., "Slicing procedure for layered manufacturing techniques," *Computer-Aided Design*, Vol. 26, No. 2, February 1994, pp. 119-126.

[Farouki94] Farouki, R.T., and Abrams, S.R., "Offset curves in layered manufacturing," *Manufacturing science and engineering*, Vol. 68, No. 2, 1994, pp. 557-568.

[Franklin90] Franklin, W.R., and Kankanhalli, M.S., "Parallel object-opace hidden surface removal," *Computer Graphics*, Vol. 24, No. 2, August 1990, pp. 87-94.

[Gaskins92] Gaskins, T., *PHIGS programming manual*, O'Reilly & Associates, Inc., Sebastopol, CA, 1992.

[Guduri92] Guduri, S., Crawford, R.H., and Beaman, J.J., "A method to generate exact contour files for Solid Freeform fabrication," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 3-5, 1992, pp. 95-101.

[Jamieson95] Jamieson, R., and Hacker, H., "Direct Slicing of CAD models for Rapid Prototyping," *Rapid Prototyping Journal*, Vol. 1, No. 2, 1995, pp. 4-12.

[Kirschman91] Kirschman, C.F., and Jara-Almonte, C.C., "A parallel slicing algorithm for solid freeform fabrication processes," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, Austin, Texas, August 12-14, 1991, pp. 26-33.

[Kulkarni95] Kulkarni, P., and Dutta, D., "Adaptive slicing for parametrizable surfaces for layered manufacturing," *proc., Design Automation Conference*, Boston, Massachusetts, Sept. 17-20, 1995, pp. 211-217.

[Naylor90] Naylor, B., Amanatides, and J., Thibault, W., "Merging BSP trees yields polyhedral set operations," *Computer Graphics*, Vol. 24, No. 2, August 1990, pp. 115-124.

[Preparata88] Preparata, F.P., and Shamos, M.I., *Computational geometry*, Springer-Verlag, New York, NY, 1988.

[Rock91a] Rock, S.J., and Wozny, M.J., "A flexible file format for Solid Freeform Fabrication," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 12-14, 1991, pp. 1-11.

[Rock91b] Rock, S.J., and Wozny, M.J.," Utilizing topological information to increase scan vector generation efficiency," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 12-14, 1991, pp. 28-36.

[Rock92] Rock, S.J., and Wozny, M.J., "Generating Topological Information from a Bucket of Facets," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 3-5, 1992, pp. 251-258.

[Sheng95] Sheng, X., and Meier, I.R., "Generating Topological Structures for Surface Models," *IEEE Computer Graphics & Applications*, Vol. 15, No. 6, November 1995, pp. 35-41.

[Suh94] Suh, Y., and Wozny, M.J., "Adaptive Slicing of Solid Freeform Fabrication Processes," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 8-10, 1994, pp. 404-411.

[Weiler86] Weiler, K.J., *Topological structures for geometric modeling*, Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY, August 1986.

[Wozny92] Wozny, M.J., "Systems issues in Solid Freeform Fabrication," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 3-5, 1992, pp. 1-15.

[Yang94] Yang, D.C.H., Jou, Y., Kong, T., and Chuang, J-J., "Laser beam diameter compensation for Helisys LOM machine," *proc., Solid Freeform Fabrication Symposium*, University of Texas at Austin, Austin, Texas, August 8-10, 1994, pp. 171-178.

Vita

I was born in Toulouse, the French capital of aeronautic technology, on January 25, 1973. From a childhood spent in western Africa, I learned individuality, tolerance, and an enthusiasm for meeting foreign cultures. I have also been interested in aeronautics for a long time. Additionally, and fortunately (or unfortunately, I do not know yet), my father bought me an Apple II for my twelfth birthday: I spent Sundays in front of a computer screen, programming computer games (technically, they were computer games), or a postprocessor for a father's finite element code (my father really pushed me for this one). I had, and still have, a deep love for Africa, which is certainly my true fatherland. However, my interest in engineering and computers prompted my return back to France and my travel to the United States, countries whose opportunities in mechanical engineering and computer science allowed me to content myself. My dream of playing in the NBA has not been fulfilled (yet), but the United States gave me the opportunity to experience yet another attractive culture and people, despite the fact that some folks here still claim to have trouble with my "darned French accent".