GEOMETRIC AND COMPUTATIONAL ASPECTS OF MANUFACTURING PROCESSES

by
Prosenjit K. Bose

School of Computer Science
McGill University, Montréal
Québec, Canada
September 1994

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Abstract

Two of the fundamental questions that arise in the manufacturing industry concerning every type of manufacturing process are:

1. Given an object, can it be built using a particular process?

2. Given that an object can be built using a particular process, what is the *best* way to construct the object?

The latter question gives rise to many different problems depending on how *best* is qualified. We address these problems for two complimentary categories of manufacturing processes: rapid prototyping systems and casting processes. The method we use to address these problems is to first define a geometric model of the process in question and then answer the questions on that model.

In the category of rapid prototyping systems, we concentrate on stereolithography, which is emerging as one of the most popular rapid prototyping systems. We model stereolithography geometrically and then study the class of objects that admit a construction in this model. For the objects that admit a construction, we find the orientations that allow a construction of the object.

In the category of casting processes, we concentrate on gravity casting and injection molding. We first model the process and its components geometrically. We then characterize and recognize the objects that can be formed using a re-usable two-part cast. Given that a cast of an object can be formed, we determine a suitable location for the *pin gate*, the point from which liquid is poured or injected into a mold. Finally, we compute an orientation of a mold that ensures a complete fill and minimizes the number of venting holes for molds used in gravity casting processes.

Résumé

Deux questions fondamentales dans l'industrie manufacturière concernant tous les procédés de fabrication sont:

1. Y a-t-il un procédé de fabrication spécifique pour un objet donné ?

2. Etant donné qu'un objet peut être construit par un procédé spécifique, quelle est la meilleure méthode de fabrication pour cet objet?

Plusieurs problèmes découlant de cette dernière question peuvent être énoncés selon la façon dont *meilleure* est définie. Ces problèmes sont considérés pour deux catégories complémentaires des procédés manufacturiers: les systèmes de conception de prototypes et les procédés de moulage. La méthode employée pour résoudre ces problèmes consiste a définir un modèle géométrique du procédé en question et à répondre aux questions liées au modèle.

Dans la catégorie des systèmes de conception de prototypes, nous nous concentrons sur la stéréolithographie, une méthode de fabrication qui gagne de plus en plus de popularité. On modèle la stéréolithographie d'une manière géométrique pour ensuite étudier la classe d'objets pour lesquelles une construction selon ce modèle est possible. Pour de tels objets, on cherche les orientations permettant une telle construction.

Dans la catégorie des procédés de moulage, nous nous concentrons sur les méthodes par gravité et par injection. D'abord, nous modelons le procédé et ses composantes géométriquement. Par la suite, les objets, qui peuvent être formés par des moules ayant deux parties et étant réutilisables, sont reconnus grâce à leurs caractéristiques. Un point d'injection satisfaisant par lequel le liquide peut être introduit ou injecté est ensuite déterminé pour ces moules. Finalement, pour les moules utilisés dans un procédé par gravité, l'orientation du moule est déterminée de sorte que le moule soit entièrement rempli avec un nombre minimal de poches d'air.

Statement of Originality

Except for the background material in Chapter 2, all the elements of this thesis should be considered original contributions to knowledge. Any algorithms, theorems or results appearing in this thesis that are the work of others are clearly indicated in the text. Finally, no assistance outside those mentioned in the acknowledgments has been received.

Acknowledgements

First, and foremost, I would like to thank my mentor, Godfried Toussaint, for making the experience of getting a Ph.D. more interesting and enjoyable than I had imagined (*a vertex on the convex hull!!*). Through his endless creativity and passion for research, he opened my eyes to the excitement and beauty in geometry. He provided me with many opportunities to learn from and work with different researchers from around the world by inviting me to his workshops and encouraging me to attend many different conferences. Above all, I thank him for his friendship.

Luck was a key behind the choice of this thesis topic. François Trochu, from the department of Mechanical Engineering at the Université de Montréal, invited Godfried one day to discuss polygon triangulations. At the end of their discussion on triangulations, Godfried casually asked François what his main field of research was. As François described his field, *Injection Molding*, several geometric problems leapt into Godfried's mind which François confirmed were important to his field. This event, coupled with several other meetings with François, gave rise to the problems addressed in Chapters 3, 4 and 5. A preliminary version of the work in Chapters 4 and 5 appears in [12] and [14], respectively. I thank François for seeking to find connections between manufacturing and geometry and subsequently bringing many interesting problems to our attention.

Chapter 6 has an interesting story. Binhai Zhu, Godfried and I started working on the problems addressed in that chapter from a GIS (Geographic Information Systems) perspective. We were joined by some visiting professors from Spain, Jesus Garcia and Goyi Blanco. Later, we discovered that Mark Overmars and his student Boudewijn Asberg had studied the same geometric problem independently from a manufacturing perspective based on some work done earlier by Gordon Wilfong. This gave rise to the eight author paper [5].

The work in Chapter 7 was initiated by Marc van Kreveld, David Bremner and myself at Godfried's workshop on manufacturing processes and completed upon our return. A preliminary version of this work appears in [15].

Marc van Kreveld deserves special thanks for always being supportive. It is always a pleasure to work with him. I thank Luc Devroye for pushing me to the limits of my ability and sometimes beyond. I thank David Avis for his encouragement and support. I thank David Bremner for the many interesting discussions we've had, for helping me out through some rough moments, and for having taken the time to proofread the thesis. I thank Sandro Mazzucato for translating the abstract. I thank Denis Therien for allowing me the privilege of his office and computer while he was away. I thank Beppe, Paco, David Eu, Eric Guevremont, Sue, Bill, Kathleen, Elsa, Hazel, Jean-Marc, Claudia and Mary for making McGill a very lively and interesting place to work in.

I would like to thank the technical staff and the secretaries, especially Lorraine, for their assistance above and beyond the call of duty. I am grateful for the financial support that I received from NSERC, FCAR, and McGill University.

I would like to thank my family for supporting me in my endeavors. Finalement, merci Linda d'être là pour moi.

To Linda and my mother,
For making it all worthwhile

Contents

List of Figures

Chapter 1

Introduction

In the manufacturing industry, there are many different types of production methods such as injection molding, gravity casting, NC machining, laser sculpting, automated welding and 3-D printing (stereolithography), available to construct an object. However, every manufacturing process imposes certain restrictions on the types of objects that can be constructed as well as the way a given object may be built. For example, a sphere cannot be built in one setup using 3-axis NC machining, but can be easily built using injection molding or gravity casting. Also, the best way of constructing a cube using stereolithography is to place it on one of its faces. This leads to two fundamental questions concerning every type of manufacturing process:

1. Given an object, can it be constructed using a particular process?

2. Given that an object can be built using a particular process, what is the *best* way to construct the object?

The latter question gives rise to many different problems depending on how *best* is qualified. The geometry of the object, coupled with the restrictions imposed by the particular manufacturing process under consideration, play a vital role in determining the answer to these questions.

The importance of these questions is quite evident. For example, when designing an object to be built by a certain type of manufacturing process, currently an engineer must always keep in mind the process used to construct the object. This limits the creativity of the engineer since the question of design feasibility must be kept in mind while creating the object. In fact, the engineer is never really quite sure whether the object can be built since no formal method exists to determine the feasibility of an object for most manufacturing processes. To resolve this problem, a practical algorithm is needed to determine, given an arbitrary object, whether or not it can be built using any of the known manufacturing processes. The benefits of such a system would be two-fold. Firstly, an engineer would have an algorithm to verify whether an object can be created using a particular manufacturing process (i.e. a type of automatic design verification). Secondly, a list of the possible manufacturing processes that can build a particular object would allow an engineer to design something and then determine which manufacturing process would be most cost efficient. In fact, two of the primary ways of reducing costs in manufacturing engineering, according to [83], are to

1. Determine whether the product as designed and developed is producible.

2. Determine the manufacturing process allowing production within product specifications at the lowest cost.

In this thesis, we address these fundamental issues for several manufacturing processes. In order to better understand the power as well as the limitations of each manufacturing process under consideration, we first develop a mathematical model of the process, and then analyze the class of objects that can be constructed under the given model. Having established that an object can be constructed, we then see what is the best way to construct the object in this model.

This approach for understanding a particular manufacturing process is not novel. Many different mathematical models of different manufacturing processes have already been studied [44]. Traditionally, models attempt to reflect the physics behind a production method. As a result, the models are fairly complicated involving differential equations, fluid dynamics, thermodynamics and so forth. In fact, many of these models form the backbone of simulation programs that simulate a particular process. Although these models may accurately reflect the particular process being modelled, they are quite complex and difficult to analyze. The novelty in our approach comes from the way we model a process. Our approach is to extract the geometric essence of the manufacturing processes we consider and answer questions from a purely geometric perspective. These results do not eradicate the need for simulations, however, our solutions are conceptually and computationally simple and provide a first approximation that may greatly reduce the time needed to find better solutions through simulation. In fact, our solutions to the problems in Chapters 3, 4, 5, and 7 provide an alternative to the trial and error approach currently used [2, 71]. The more vital the role that the geometry of the problem plays, the better our solutions are.

We present relatively simple discrete geometric models of the processes under investigation. The objects we study are polygons and polyhedra. These objects can be handled by almost all CAD/CAM systems [1, 8, 9]. Of the diverse manufacturing processes, only NC machining has been studied extensively from this perspective [42]. We investigate problems concerning manufacturing processes that fall into two different but related categories.

The first category of manufacturing processes comprises rapid prototyping systems (see [6] for details). As suggested by the name of this category, all manufacturing processes used to build prototypes fall into this category. These systems are used in the design phase. While designing an object, these systems can be used to produce three dimensional prototypes of a given object which provide much more information to the designer, as well as to the eventual producer of the object, than do two dimensional drawings of the object. Rapid prototyping systems have been gaining more importance in recent years since this technology is becoming affordable and saving companies such as GM, FORD and IBM millions of dollars. Currently, rapid prototyping is an $8 billion per year business and demand is growing at 80 to 90 per cent a year [36]. Stereolithography is emerging as the dominant process in this category. According to Marshall Burns (physicist, consultant and author of 'Automated Fabrication: Improving Productivity in Manufacturing', published by PTR Prentice-Hall Inc.), stereolithography is going to start a revolution in the manufacturing industry and in 20 to 25 years will be as common as computer printers. In Chapter 6, we study, from a geometric perspective, the powers as well as the limitations of stereolithography by characterizing the objects that can be constructed by stereolithography.

Once the design of an object has been completed, the next step is production. The second category of manufacturing processes we study entails casting processes, which has always been one of the most popular methods used to mass produce objects [71, 31, 47, 81, 32]. Basically, a casting process is a manufacturing process that uses a mold or cast to produce an object. A mold or cast, as defined in [17], refers to the whole assembly of parts that make up a cavity into which liquid is poured to give the shape of the desired component when the liquid hardens. The importance of this category is evident since many of the objects we see everyday such as cups, forks, door knobs, and most plastic objects are built using casting processes. The processes that we study from this category fall into two main groups: injection molding and gravity casting. Each of the two methods produces an object by filling a mold or cast of the given object with a liquid, and removing the object once the liquid has hardened (see Figure 1.1). The difference between the two methods is that liquid is injected using pressure into the mold in injection molding processes whereas liquid is poured into the mold and gravity is the sole force acting on the liquid in gravity casting processes [32].

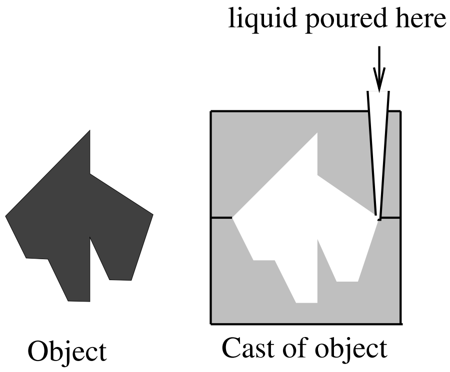

Figure 1.1: An object and its cast.

In Chapter 7, we study the problem of determining given an object, modelled by a simple polyhedron, whether or not a two-part cast of the object can be made. Currently, the two-part cast is the most popular type of cast used in manufacturing. Casts consisting of more than two parts are difficult to produce and are not as efficient to use as the two-part casts.

If a cast of a prototype object can be made, a cast designer is then faced with the problem of determining a suitable location for the pin gate. The *pin gate* is the point on the mold from which the liquid is poured or injected into the cavity. The location of the pin gate plays an important role in determining whether or not an object built by one of the two manufacturing processes will have many surface defects. Many factors play a role in determining a suitable location for the pin gate. In Chapter 3, we analyze the different geometric factors involved in the location of a suitable pin gate and present algorithms for determining pin gate locations satisfying certain geometric criteria.

Once a cast of an object has been built, finding a favorable orientation of the mold that minimizes surface defects and allows the most complete fill also becomes a challenge. We model the geometric aspects of the filling of a mold for gravity casting and determine an orientation that minimizes the number of venting holes and allows the most complete fill. Chapters 4 and 5 concentrate on this problem for molds modelled as simple polygons and simple polyhedra, respectively.

The chapters in this thesis are not ordered in the sequence presented here, but they are ordered such that geometric tools and techniques developed in one chapter can be used in a later one.

Chapter 2

Notation and Preliminaries

In this chapter, we review some of the notation and terminology of this thesis. Notation and terminology specific to a particular chapter will be introduced in that chapter. For more detailed definitions, the reader can refer to O'Rourke [64], or Preparata and Shamos [69].

The model of computation assumed throughout the thesis is the *real* RAM. In the *real* RAM model, each storage location is capable of holding a single real number, and the following operations are primitive and available at unit cost (or unit time):

1. Arithmetic operations.
2. Comparisons between two reals.
3. Indirect addressing of memory.
4. Square roots.

A *simple polygon* P is a simply connected subset of the plane whose boundary is a closed chain of line segments. A polygon P is denoted by a set of vertices $v_1, v_2, \ldots, v_{n-1}, v_n$ such that each pair of consecutive vertices is joined by an edge, including the pair $\{v_n, v_1\}$. Unless stated otherwise, the vertices are assumed to be in clockwise order, so that the interior of the polygon lies to the right as the boundary of the polygon is traversed.

The open interior of the polygon P is denoted by $int(P)$, the boundary by ∂P, and the open exterior by $ext(P)$. The boundary is considered part of the polygon; that is, $P = int(P) \cup \partial P$.

Given a line segment e, the line containing e is denoted by $L(e)$. A *convex edge* of a simple polygon refers to an edge e where both endpoints of e are convex vertices. Similarly, a *reflex edge* of a simple polygon refers to an edge e where both endpoints of e are reflex vertices.

Given two points a and b in the plane, $[ab]$ and (ab) denote respectively the closed and open line segments between the two points. A *chord* of a polygon is a line segment between two points on the polygon boundary such that the open line segment is contained in the interior of the polygon. A chord divides a polygon into two subpolygons.

We define a *simple polyhedron* P as in O'Rourke [64]. The boundary of P is a finite collection of planar, bounded convex polygonal faces such that

1. The faces are disjoint or intersect *properly*. (A pair of faces intersect properly if either they have a single vertex in common or have two vertices, and the edge joining them, in common.)
2. The *link* of every vertex is a simple polygonal chain. (Triangulate the faces that have vertex v on their boundary. The link of v is the collection of edges opposite v in all the triangles incident to v.)

3. The *one-skeleton* is connected. (The *one-skeleton* is the graph of edges and vertices of the polyhedron.)

The boundary is closed and is denoted as ∂P. The boundary encloses a bounded region of space, denoted as $int(P)$. The polyhedron consists of the boundary and its interior, (i.e. $P = int(P) \cup \partial P$). The (unbounded) *exterior* of P is denoted as $ext(P)$. As this thesis only deals with simple polyhedra, we will refer to them as polyhedra in the remainder of the thesis. The vertices and the edges of the faces are the *vertices* and the *edges* of the polyhedron. The open interior of the faces are called the *facets* of the polyhedron. Therefore, for a facet f, the closure of the facet is the face and denoted $cl(f)$.

For two polyhedra P and Q whose interiors lie on different sides of a plane h, and which are both bounded by the same facet f that lies inside h, we define the *union* of P and Q as the polyhedron with all vertices of P and Q, with all facets of P and Q except f, and with all edges of P and Q except the ones contained in h that bound two parallel facets.

The intersection of a polyhedron with an arbitrary plane results in a collection (possibly empty) of simple polygons (or line segments or points) lying on the plane. A polygon in this collection will be referred to as a *sectional* polygon. Notice that a sectional polygon divides the polyhedron into two simple polyhedra. Thus in this sense a sectional polygon is the three dimensional equivalent to a chord in a polygon.

By a *direction* we mean an equivalence class of *oriented* parallel lines. A given direction Θ will be specified by a point on a unit circle in the following way. Let C be a unit circle with center o. Let x be a point on the boundary of the circle such that $ray(ox)$ is parallel to and with the same orientation as Θ. Then direction Θ is represented by the point x. (Refer to Figure 2.1). A point that is diametrically opposite to x on the unit circle represents the *inverse or opposite direction* to Θ and is denoted by $opp(\Theta)$. A *right (left) normal* to a given direction Θ is an equivalence class N of *oriented* parallel lines with the property that every member of N is orthogonal to Θ and oriented to the right (left) of Θ. The right normal of Θ will be denoted by $N^+(\Theta)$ and the left normal will be denoted by $N^-(\Theta)$.

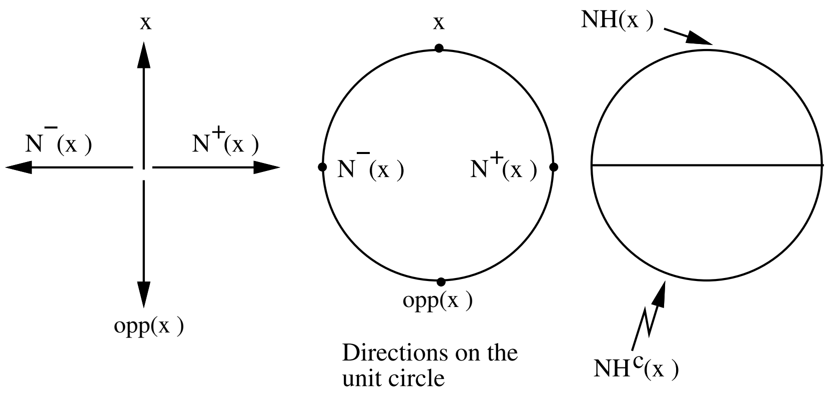

Figure 2.1: Illustrating the representation of directions.

An equivalence class of parallel lines H will be specified by a pair of points p_1 and p_2 that are diametrically opposite on the unit circle, such that the line determined by the two points is parallel to a line in H. A *normal* to a line in H is an equivalence class N of parallel lines with the property that every member of N is orthogonal to H. Given an oriented direction Θ, we define the *normal* to Θ, denoted by $N(\Theta)$, as the equivalence class of parallel lines that are orthogonal to Θ. Notice that the two points representing the normal divide the boundary of the unit circle into two semi-circles. We refer to the open semi-circle containing the point representing the direction Θ as the open *normal semi-circle* or the open *normal half-plane* of Θ and denote it as $NH(\Theta)$. The closed semi-circle is denoted by $NH[\Theta]$. The open semi-circle not containing the point representing the direction of Θ will be denoted by $NH^c(\Theta)$. Similarly, the closed semi-circle not containing the point is $NH^c[\Theta]$.

Given two points a and b on the unit circle, let $arc[a,b]$ and $arc(a,b)$ denote respectively the closed and open arcs of the unit circle from a to b in the clockwise direction.

Similarly, we represent the set of all directions in 3-space by the points on the surface of a unit sphere (see Figure 2.2 for definitions to follow). Let $\mathcal{S}$ be the unit sphere centered at the origin o. Any point p on $\mathcal{S}$ represents the direction $\overrightarrow{op}$. A point that is diametrically opposite to p on the unit sphere represents the *inverse or opposite direction* to direction p and is denoted by $opp(p)$. Notice that all the points s on the boundary of $\mathcal{S}$ with the property that $\overrightarrow{os} \cdot \overrightarrow{op} = 0$ ($\cdot$ represents the inner product) form a great circle. We denote this great circle by $N(p)$ since all these points are directions that are orthogonal to p. The great circle $N(p)$ divides the sphere into two half-spheres. The open half-sphere containing p will be denoted as $NH(p)$ and the closed half-sphere by $NH[p]$. The open and closed half-spheres not containing p will be denoted by $NH^c(p)$ and $NH^c[p]$, respectively. When considering the angle between two vectors, we always mean the smaller angle unless stated otherwise.

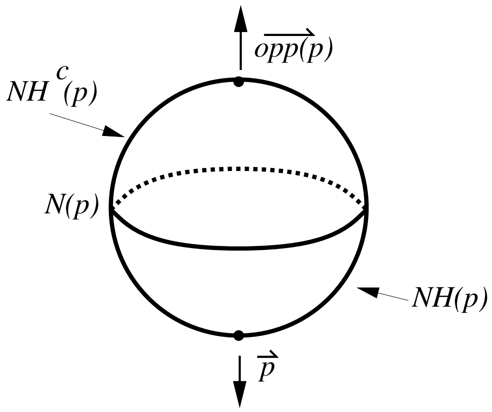

Figure 2.2: The sphere of directions.

For a non-vertical plane h, we denote by h^+ and h^- the open half-spaces above and below h, and by $cl(h^+)$ and $cl(h^-)$ the closed half-spaces above and below h . If h is vertical but does not contain a line parallel to the y-axis, then h^+ and h^- denote the open half-spaces bounded by h that contain the points $(0, \infty, 0)$ and $(0, -\infty, 0)$, respectively. If h is vertical and contains a line parallel to the y-axis then h^+ and h^- denote the open half-spaces bounded by h that contain the points $(\infty, 0, 0)$ and $(-\infty, 0, 0)$, respectively.

Chapter 3

Pin Gate Location

3.1 Introduction

In this chapter, we consider the problem of determining a suitable location for the *pin gate*. The *pin gate* is the point on the mold from which the liquid is poured or injected into the cavity. The location of the pin gate plays an important role in determining whether or not an object built by one of the two manufacturing processes will have surface defects. Many factors play a role in determining a suitable location for the pin gate when considered from the point of view of fluid dynamics and physics of the whole molding process. To date, trial and error, guided by engineering experience, has been the main method in determining a suitable location for the pin gate [47], [71], [86]. However, through this experience, a few of the key characteristics of an ideal location for a pin gate have been uncovered.

> If the distance from the gate to the extremities of the mold cavity is too great, the metal freezes prematurely, and misruns result. [47]

This quote points out one of the key problems faced by cast designers. In order to avoid this problem, designers must place the pin gate at a location where the distance from it to the extremities of the mold cavity is not too great. Another key characteristic of casts that leads to surface defects is the presence of many "sharp corners or overhanging or protruding sections..." [2]. These "sharp corners" disrupt the flow of molten liquid leading to surface defects. Therefore, the pin gate must be placed in a location such that the flow of molten liquid from the gate does not encounter too many sharp corners or make too many turns. For an overview of the many other factors causing defects in molds and casts, the reader is referred to [47], [2].

These observations allow one to deduce the following properties for a good location for a pin gate:

Property 1: The maximum distance from the pin gate to any point in the object should be small.

Property 2: The maximum number of turns the liquid takes on its path from the pin gate to any point in the object should be small.

When viewed from a purely geometric perspective, these problems can indeed be solved optimally. The geometric solutions provide an initial approximation that can aid in the search for a suitable location. In this chapter, we solve the pin gate location problem for molds modelled as simple polygons which find applications in polymer molds. In practice, many 3-dimensional objects are almost flat so that in effect they can be considered as 2-dimensional. Therefore the 2-dimensional theory is more important than may appear at first glance, and sheds some light on the 3-dimensional problem.

The two properties that a pin gate should satisfy have several geometric interpretations. Property 1 can be interpreted as the point inside the simple polygon whose maximum distance to any point in the object is minimized. If distance is measured in the Euclidean metric, this point is referred to as the constrained Euclidean center. Sometimes a pin gate is constrained to lie on the boundary of the mold. In such a case, Property 1 can be interpreted as the point on the boundary of the simple polygon whose maximum distance to any point in the polygon is minimized with respect to all points on the boundary. This point is referred to as the *boundary-constrained* Euclidean center. On the other hand, distance can be measured by the geodesic metric, i.e., the minimum distance the liquid must travel inside the mold to reach a destination. In this case, Property 1 places the pin gate at the geodesic center, which by definition is constrained to lie inside the polygon, and the *boundary-constrained* geodesic center, respectively.

Property 2 can be interpreted as the link metric. The link metric measures the number of turns in a path between two points. For example, if two points can be joined by a line segment, then they are at link distance 1. The points inside a simple polygon, whose link distance to any other point in the polygon is minimized, are referred to as the link center. If the pin gate is constrained to the boundary, then it is referred to as the *boundary-constrained* link center.

3.2 Constrained Euclidean Center

In this section, we show how to find the point inside a simple polygon P as well as the point on ∂P whose maximum Euclidean distance to every point of P is minimized. These points are known as the *Euclidean center constrained to lie in the polygon*, and the *Euclidean center constrained to lie on the boundary of the polygon*, respectively.

We first review the problem of finding the Euclidean center. Given a set S of n points in the plane, the Euclidean center is the center of the smallest circle enclosing the points of S. This problem has a rich history. We summarize as in [69]. The search for an efficient algorithm seems to have begun in 1860 by Sylvester [82]. Later, Rademacher and Toeplitz [72] noted that the smallest enclosing circle is unique and is either the circumcircle of three points of the set or defined by a diametrical pair. This immediately gives an $O(n^4)$ algorithm. Elizinga and Hearn [33, 34] improved this to $O(n^2)$. Much work was done from an Operations Research perspective by viewing the problem as a minimax facility location problem, where the Euclidean center is the point whose greatest distance to any point of the set is minimized [41, 84, 48]. An $O(n \log n)$ time solution to this problem was proposed by Shamos and Hoey [77], but Bhattacharya and Toussaint [10] pointed out some errors in [77] and subsequently proposed an alternate $O(n \log n)$ time solution. Preparata [66] and Melville [60] also proposed an alternate $O(n \log n)$ time solution. However, no $\Omega(n \log n)$ time lower bound for the problem was known. A search for a resolution to this problem ensued, culminating in the discovery of an elegant $\Theta(n)$ time solution to the problem by Megiddo [57].

The Euclidean center of the vertices of a simple polygon may be a good candidate for the location of the pin gate, but the center might lie outside the polygon (see Figure 3.1). Therefore, the location of the center must be constrained to lie inside the polygon or on its boundary since otherwise it cannot serve as a pin gate. Therefore, given an object modelled as a simple n vertex polygon, we wish to find the point lying inside the polygon whose maximum Euclidean distance to any point is minimized with respect to all points in the polygon. Since the furthest neighbor of a point must be a vertex, we can restrict our attention to finding the point lying inside the polygon

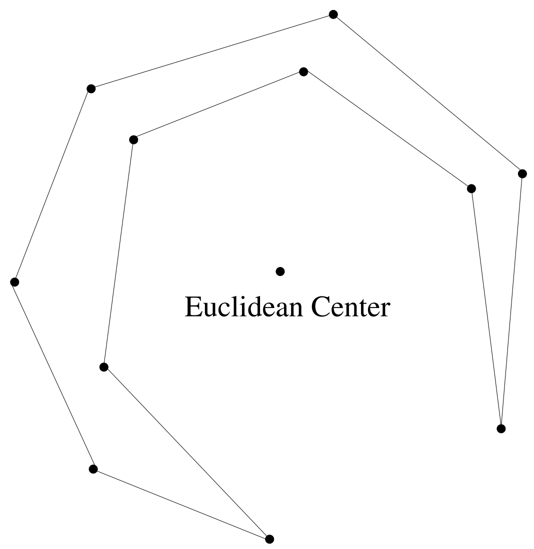

Figure 3.1: Euclidean center outside polygon.

whose maximum Euclidean distance to any vertex is minimized with respect to all points in the polygon. We also want the point on the boundary whose maximum Euclidean distance to any vertex is minimized with respect to all points on the boundary. Although the Euclidean center is unique, the Euclidean center constrained to lie inside the polygon as well as the Euclidean center constrained to lie on the boundary of the polygon need not be unique, as depicted in Figure 3.2.

3.2.1 Center Constrained to a Polygonal Region

We solve a slightly more general problem than the one mentioned in the introduction. Suppose we are given a set $S = \{s_1, s_2, \ldots, s_k\}$ of k points (in general position) in the plane E^2, and an n vertex simple polygon P. We wish to find the point c in P whose maximum distance to any point in S is minimized. If c is not constrained to lie in P, then it is the Euclidean center of S. However, we refer to c as the Euclidean center of S constrained to P and denote it by $EC_P(S)$.

Our algorithms make use of the *furthest point Voronoi diagram* of the set S, denoted as $FPVD(S)$. Given a point $x \in E^2$, we let $\phi(x)$ denote the furthest neighbors of x in S, that is the set of points in S such that $d(x, \phi(x)) = \max_{y \in S} d(x, y)$ where d is the Euclidean distance function. The $FPVD(S)$ partitions the plane into unbounded convex cells, $V(s_i)$, such that for any point $p \in V(s_i)$, $s_i \in \phi(p)$. This structure can be computed in $O(n \log n)$ time [69]. A list of the many geometric properties of the furthest point Voronoi diagram can be found in [62, 69, 10].

We first review some properties of the Euclidean center which will help us in finding its constrained counter-part.

Lemma 3.2.1 *[69, 10] The Euclidean center of S lies on the midpoint of the diameter of the set S, $DIAM(S)$, provided that the circle with $DIAM(S)$ as diameter contains the set S.*

Lemma 3.2.2 *[69, 10] If the Euclidean center does not lie on the midpoint of $DIAM(S)$, then it lies on the vertex of the $FPVD(S)$ that yields the smallest spanning circle.*

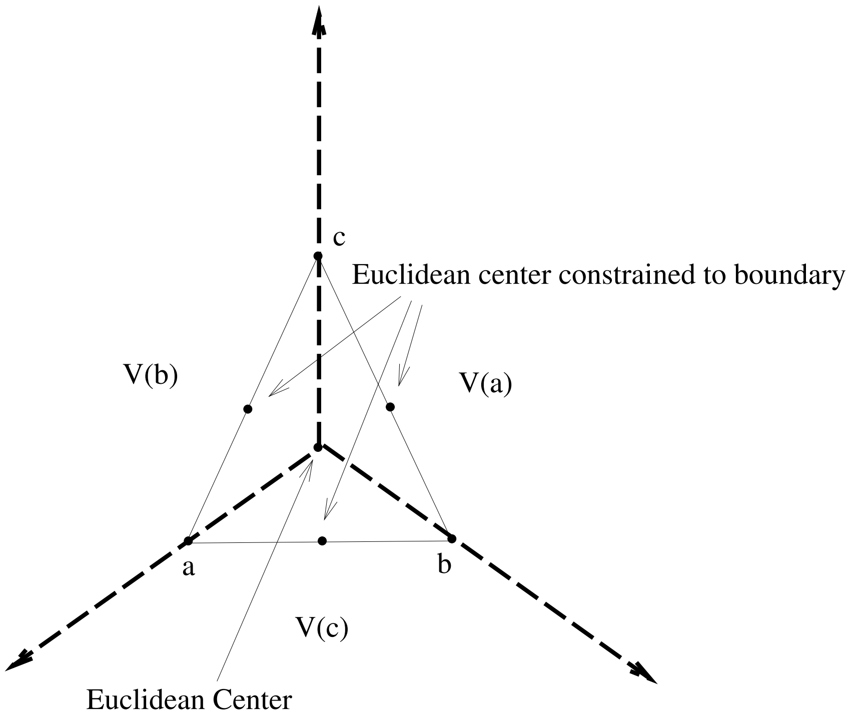

Figure 3.2: Constrained Euclidean center may not be unique.

These two lemmas characterize the location of the Euclidean center. When considering the constrained version of the problem, notice that if the Euclidean center happens to lie inside the constraining polygon, then it is also the constrained Euclidean center. However, difficulties arise when the Euclidean center does not lie inside the polygon. These difficulties are resolved in the following lemmas.

Lemma 3.2.3 *The Euclidean center of S constrained to lie in P is the midpoint of $DIAM(S)$ provided that the diametral circle contains the set S, and the midpoint is contained in P.*

Proof: Follows from Lemma 3.2.1. ∎

Before tackling the problem of determining the location of $EC_P(S)$ when it is not on the midpoint of $DIAM(S)$, we first establish a lemma that will prove useful. Let a, b be two points in S such that both a and b are on the convex hull of S, $[ab]$ is not the diameter of S, and $V(a)$ and $V(b)$, the two cells of $FPVD(S)$ representing a and b, respectively, are adjacent and separated by an edge e. Let x be a point on the interior of e, and let $\epsilon > 0$ be any small constant.

Lemma 3.2.4 *There exists a point $y \in e$ with $d(x, y) < \epsilon$ such that $d(y, a) < d(x, a)$ and $d(y, b) < d(x, b)$.*

Proof: See Figure 3.3. The edge e must lie on the bisector of line segment $[ab]$, since the points on e are equidistant from both a and b. The points a, b, x must form a triangle because otherwise $[ab]$ would be the diameter. Since x is contained in $int(e)$, let y be a point on e in $\triangle(abx)$ such that $d(x, y) < \epsilon$. The lemma follows. ∎

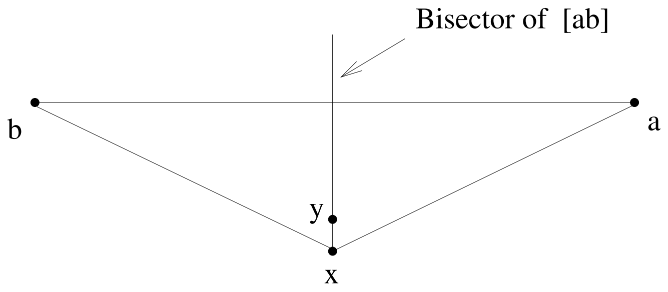

Figure 3.3: Illustration for proof of Lemma 3.2.4.

Lemma 3.2.5 *If $\|S\| > 1$ then a point b of S cannot lie in $V(b)$.*

Proof: Let $x \in S$ be a point distinct from b. Note that $d(b, b) = 0$, however, $d(b, x) > 0$ which contradicts the fact that $b \in V(b)$. ∎

We now complete the characterization of $EC_P(S)$.

Lemma 3.2.6 *If the Euclidean center of S constrained to lie in P is not the midpoint of $DIAM(S)$, then it lies on one of the following points that yields the smallest spanning circle:*

1. *a vertex of the $FPVD(S)$ contained in P,*
2. *a proper intersection point of the $FPVD(S)$ and the boundary P,*
3. *a vertex of the polygon P,*
4. *a point x on an edge e of P with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d(y, \phi(x)) \geq d(x, \phi(x))$.*

Proof: If $EC_P(S)$ does not lie on any of the points mentioned in the statement of the lemma, then it must lie in one of the regions described in the following four cases. We show that each of these cases leads to a contradiction. For simplicity of exposition, let $c = EC_P(S)$.

Case 1: c is a point in the interior of a cell of the $FPVD(S)$, and in $int(P)$. Let $V(b)$ be the cell containing c. By the Jordon Curve Theorem [64], line segment $[bc]$ must intersect ∂P or $V(b)$ since $b \notin V(b)$ by Lemma 3.2.5. Let x be the intersection point closest to c. The point x must be in $V(b)$. Therefore the circle centered at x with radius $d(x, b)$ encloses the set S. However, $d(x, b) < d(c, b)$ by construction. Hence, we have a contradiction.

Case 2: c is a point in the interior of a cell of the $FPVD(S)$, and in the interior of an edge e of P but does not satisfy the property that $\forall y \in e$, if $\phi(y) = \phi(c)$ then $d(y, \phi(c)) \geq d(c, \phi(c))$. Since the latter property is not satisfied, a point $x \in e$ such that $\phi(x) = \phi(c)$ and $d(x, \phi(c)) < d(c, \phi(c))$ must exist. However, the very existence of x contradicts that c is the constrained Euclidean center since the circle centered at x with radius $d(x, \phi(c))$ encloses S.

Case 3: c is a point in the interior of an edge e of the $FPVD(S)$, and in $int(P)$. Let $V(a)$ and $V(b)$ be the two cells separated by the edge e. Since c is not on the diameter of the set S, by Lemma 3.2.4 we know that there exists a point x in e and in $int(P)$ such that $d(x, a) < d(c, a)$ and $d(x, b) < d(c, b)$. This contradicts that c is the constrained Euclidean center.

Case 4: c is a point in the interior of an edge e_v of the $FPVD(S)$, and in the interior of an edge e_p of P such that e_v and e_p intersect but not properly. Same argument as Case 3.

■

Lemma 3.2.3 and Lemma 3.2.6 characterize the location of $EC_P(S)$. We outline the following algorithm to compute this point.

Algorithm 1: *Euclidean Center of P constrained to lie in S*

Input: A set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a simple polygon $P = \{p_1, p_2, \ldots, p_n\}$.

Output: $EC_P(S)$

1. Compute the $FPVD(S)$.
2. Compute $DIAM(S)$.
3. Compute the circle C having $DIAM(S)$ as diameter.
4. Preprocess P in $O(n \log n)$ time for point inclusion testing in $O(\log n)$ time using the algorithms of Kirkpatrick [45] or Sarnak and Tarjan [75].
5. If the midpoint of C is contained in P and all the points of S are contained in C then exit with the midpoint of $DIAM(S)$.
6. Compute the set of vertices of $FPVD(S)$ contained in P. Let V_c represent this set.
7. Compute the set of intersections $I_c = \{i_1, i_2, \ldots, i_k\}$ of P with $FPVD(S)$.
8. Partition each edge e_i of P such that for every pair of points $x, y \in e_i$, we have that $\phi(x) = \phi(y)$. Denote the j^{th} partition of e_i by e_{ij}.
9. For each e_{ij}, compute the point on e_{ij} closest to $\phi(e_{ij})$. If this point is not an endpoint of e_{ij}, place it in the set E_c.
10. Let P_c represent the vertices of P. For each point c in V_c, I_c, P_c, E_c, compute the smallest spanning circle with center c. Let SP represent this set.
11. Select all the smallest circles in SP, and output their centers and the radius.

Notice that we assumed that the number of vertices of P equals the number of points in S. Clearly, this need not be the case, however, this assumption simplifies the complexity of notation. It is straightforward to repeat the complexity analysis when P and S have different cardinalities.

Theorem 3.2.1 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a simple polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie in P in time $O(n \log n + k)$ where n is the size of the input and k is the number of intersections between the edges of the $FPVD(S)$ and P.*

Proof: The correctness of the algorithm follows from Lemmas 3.2.3 and 3.2.6.

Let us analyze the complexity of the algorithm. Step 1 of the algorithm can be computed in $O(n \log n)$ time using the algorithm of Shamos [69]. Step 2 can be computed in $O(n \log n)$ time by first computing the convex hull of S and then finding the diameter of the convex hull. Preprocessing for point inclusion can be done in $O(n \log n)$ using the algorithm of Kirkpatrick [45] or Sarnak and Tarjan [75]. Step 5 can be achieved in $O(n \log n)$ time by using the point inclusion test. Step 6 can be done in $O(n \log n)$ time using the point inclusion test. Step 7 can be computed in $O(n \log n + k)$

time where k is the number of intersections between P and $FPVD(S)$ using the algorithm of Chan [18]. If we color the segments in $FPVD(S)$ blue and the edges of P red, then the algorithm of [18] reports the intersections along each edge of P in sorted order. Once these intersection points have been computed, Step 8 and 9 can be achieved in $O(n+k)$ time. Step 10 can be computed in $O(n+k)$ time since it takes constant time to compute the circle and there are $O(n+k)$ points in the set SP. Finally, Step 11 can be computed in $O(n+k)$. Therefore, the total complexity of the algorithm is $O(n \log n + k)$ time. ∎

For simple polygons, k can be $O(n^2)$, however, for convex polygons, we notice the following: a line segment can intersect a convex polygon only twice. Therefore, since $FPVD(S)$ consists of $O(n)$ line segments, there can only be $O(n)$ intersections between $FPVD(S)$ and an n vertex convex polygon. Therefore, we have:

Corollary 3.2.1 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a convex polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie in P in time $O(n \log n)$ where n is the size of the input.*

3.2.2 Center Constrained to a Polygonal Chain

With a slight modification, Algorithm 1 can compute the Euclidean center constrained to lie on the boundary of the polygon, denoted as $EC_{\partial P}(S)$. These modifications are outlined below.

Lemma 3.2.7 *The Euclidean center of S constrained to lie on the boundary of P is the midpoint of $DIAM(S)$ provided that the diametral circle contains the set S, and the midpoint is on the boundary of P.*

Proof: Follows from Lemma 3.2.1. ∎

Lemma 3.2.8 *If the Euclidean center of S constrained to lie on the boundary of P is not the midpoint of $DIAM(S)$, then it lies on one of the following points that yields the smallest spanning circle:*

1. *a vertex of the $FPVD(S)$ on the boundary of P,*
2. *an intersection point of the $FPVD(S)$ and the boundary P,*
3. *a vertex of the polygon P,*
4. *a point x on an edge e of P with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d(y, \phi(x)) \geq d(x, \phi(x))$.*

Proof: If $EC_{\partial P}(S)$ does not lie on any of the points mentioned in the statement of the lemma, then it must lie in one of the regions described in the following four cases. We show that each of these cases leads to a contradiction. For simplicity of exposition, let $c = EC_{\partial P}(S)$.

Case 1: c is a point in the interior of a cell of the $FPVD(S)$, and in $int(P)$. This cannot happen since c must be on the boundary of P.

Case 2: c is a point in the interior of a cell of the $FPVD(S)$, and in the interior of an edge e of P but does not satisfy the property that $\forall y \in e$, if $\phi(y) = \phi(c)$ then $d(y, \phi(c)) \geq d(c, \phi(c))$. Since the latter property is not satisfied, a point $x \in e$ such that $\phi(x) = \phi(c)$ and $d(x, \phi(c)) < d(c, \phi(c))$ must exist. However, the very existence of x contradicts that c is the constrained Euclidean center since the circle centered at x with radius $d(x, \phi(c))$ encloses S.

Case 3: c is a point in the interior of an edge of the $FPVD(S)$, and in $int(P)$. Again, c cannot lie in $int(P)$ since it is constrained to the boundary.

Case 4: c is a point in the interior of an edge e_v of the $FPVD(S)$, and in the interior of an edge e_p of P such that e_v and e_p intersect but not properly. Let $V(a)$ and $V(b)$ be the two cells separated by the edge e_v. Since c is not on the diameter of the set S, by Lemma 3.2.4 we know that there exists a point x in e_v and in e_p such that $d(x,a) < d(c,a)$ and $d(x,b) < d(c,b)$. This contradicts that c is the constrained Euclidean center.

■

Lemma 3.2.7 and Lemma 3.2.8 characterize the location of $EC_{\partial P}(S)$. The modifications to Algorithm 1 for computing these points are straightforward. Therefore, we conclude with the following.

Theorem 3.2.2 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a simple polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie on the boundary of P in time $O(n \log n + k)$ where n is the size of the input and k is the number of intersections between the edges of the $FPVD(S)$ and P.*

Corollary 3.2.2 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a convex polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie on the boundary P in time $O(n \log n)$ where n is the size of the input.*

3.3 Constrained Geodesic Center

Both versions of the constrained Euclidean center serve as good first approximations for the locations of the pin gate. However, in some cases the constrained Euclidean center may not be a point satisfying Property 1, as intended (see Figure 3.4). In fact, it may be quite bad in the sense that the liquid may have to travel quite far despite the fact that the pin gate is located at the constrained or boundary-constrained Euclidean center. The reason is that the Euclidean distance of the pin gate to all the points may not be a good measure of the actual distance the liquid must travel inside the polygon. For example, in Figure 3.4, the Euclidean center, constrained Euclidean center and boundary-constrained Euclidean center all lie on the same vertex indicated on the polygon. However, the distance that the liquid must travel inside the polygon from that point to vertex v is quite large compared to vertex c. Although for convex or "near" convex objects, the Euclidean metric may be good, it seems that the geodesic metric may serve as a better approximation since liquid is travelling inside the polygon.

In the geodesic metric, the distance between two points inside a simple polygon is defined as the length of the shortest path connecting the two points inside the polygon. The *geodesic center* of a simple polygon is the point whose maximum geodesic distance to any other point in the polygon is minimized. Therefore, by definition, the geodesic center of a simple polygon lies inside the polygon. Although the geodesic center and boundary-constrained geodesic center may serve as better approximations for the location of a pin gate, computing both centers is more difficult than their Euclidean counter-parts as we shall see.

3.3.1 Geometric Properties

The problem of computing the geodesic center of a simple n vertex polygon P, denoted $GC(P)$, was first tackled by Asano and Toussaint [7]. They gave an $O(n^3 \log\log n)$ time algorithm for

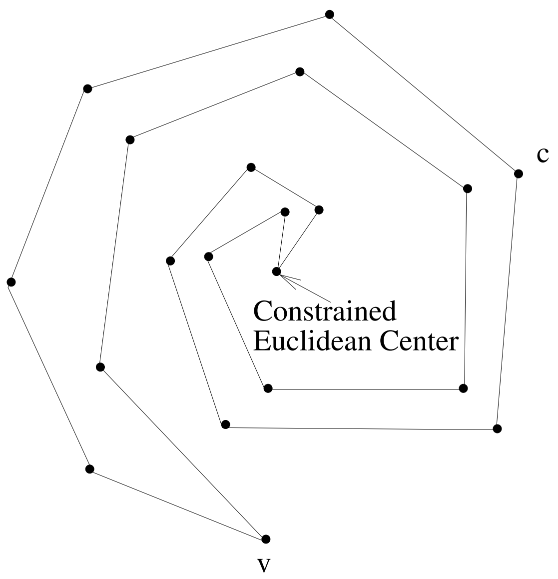

Figure 3.4: Constrained Euclidean center may not be a good approximation for the best pin gate location.

computing the center. In [7], it is shown that the geodesic center is unique and located on a vertex of the *geodesic furthest point Voronoi diagram* of P, denoted $GFPVD(P)$. The $GFPVD(P)$, like its Euclidean counter-part, divides the polygon P into cells $V(v_i)$, such that the locus of points in $V(v_i)$ is further from v_i than any other vertex of P (with distance measured with the geodesic metric). Later, Pollack, Rote and Sharir [65] reduced the complexity of computing the geodesic center to $O(n \log n)$ time. They used a different approach and achieved their time bound by a modification of Meggido's technique. Recently, Aronov et al.[3] presented an $O(n \log n)$ time algorithm for computing the $GFPVD(P)$, thus providing an alternate $O(n \log n)$ time solution for computing the center. Therefore, to compute the geodesic center of a simple polygon, any one of the above algorithms may be used, however, all of these algorithms are complicated and involved.

The problem of computing the boundary-constrained geodesic center of a simple polygon P, denoted as $GC(\partial P)$, has not previously been addressed. We concentrate on solving this problem. Like its Euclidean counter-part, the geodesic center constrained to the boundary is not necessarily unique, and not necessarily an intersection point of $GFPVD(P)$ and P. Figure 3.2 shows an example of this. If an algorithm for computing the geodesic center already exists, the following heuristic may serve as a good approximation of the boundary-constrained geodesic center.

Heuristic 3.3.1 *A heuristic for computing the boundary-constrained geodesic center is to compute the point on the boundary closest to the geodesic center.*

In some cases, this heuristic actually gives the boundary-constrained geodesic center, as seen in Figure 3.2. In the next section, we present an $O(n \log n)$ time algorithm to compute the boundary-constrained geodesic center exactly. The main idea behind the algorithm is the following. We divide the polygon boundary into polygonal chains such that the geodesic furthest neighbor of any point on a given chain is the same. Then, we compute for each chain, the point, which we call the candidate for that chain, whose distance to the furthest neighbor is the smallest compared to any other point on the chain. We select the smallest candidates as the geodesic center constrained to the boundary. We modify an algorithm of Suri [79], similar to [3], to compute this.

Given two points a, b in a polygon P, there is a unique geodesic path connecting a, b in P. We denote this path by $\pi(a, b)$ and its length by $d_G(a, b)$. Since geodesic distance is a metric, the triangle inequality holds. Therefore, we have that $d_G(x, y) \leq d_G(x, z) + d_G(z, y)$ for every three points x, y, z in P. The *geodesic furthest neighbors* of a point x in P, denoted by $\phi(x)$, are the set of points y in P such that $d_G(x, y) = \max_{\forall\ z \in P}\{d_G(x, z)\}$. Asano and Toussaint [7] showed that the geodesic furthest neighbor of a point is always a convex vertex of the polygon. The *geodesic diameter* of a polygon P, denoted as $GDIAM(P)$, is determined by the pair of points in P whose geodesic distance is maximum over all pairs of points in P. If two shortest paths do not share a point, we say they are *disjoint*; otherwise, we say that the paths *intersect*.

An important property of geodesics, at the heart of the algorithm, is the *Crossing Property* described in the following lemma.

Lemma 3.3.1 (Crossing Property) *[79] Let p_1, p_2, p_3, p_4 be four points in this order on the boundary of P. Suppose that $p_3 \in \phi(p_2)$ and $p_4 \in \phi(p_1)$. Then we also have $p_3 \in \phi(p_1)$ and $p_4 \in \phi(p_2)$.*

To compute the boundary-constrained geodesic center, we first compute a constrained geodesic decomposition of the boundary of polygon P, which is a decomposition of the boundary of P into polygonal chains $(c_1, c_2, \ldots, c_t)$ such that $\bigcup_{i=1}^{t} c_i = P$ and for every $x, y \in c_i$, $\phi(x) = \phi(y)$. We denote this decomposition as ∂-$CGD(P)$. Given this decomposition, the constrained geodesic center can be easily identified, as shall be shown in the next section. The crossing property is the key behind the algorithm. It suggests a divide-and-conquer approach to solving the problem of computing the constrained geodesic decomposition of ∂P. We first consider a restricted version of the decomposition problem, and then we show how to use its solution to compute the whole decomposition.

3.3.2 Restricted Geodesic Decomposition

The restricted version of the decomposition problem is described as follows. Let $U = (u_a, \ldots, u_b)$ be the counterclockwise chain from point u_a to point u_b on the boundary of P. Let $V = (v_c, \ldots, v_d)$ be the clockwise chain from v_c to v_d on the boundary of P, such that both chains are disjoint except possibly at the endpoints. The set of points which are the furthest neighbors of x restricted to V is denoted by $\phi_V(x)$. We want to decompose U into polygonal chains $(c_1{}', c_2{}', \ldots, c_s{}')$ such that $\bigcup_{i=1}^{s} c_i{}' = U$ and for every $x, y \in c_i{}'$, $\phi_V(x) = \phi_V(y)$. We refer to this decomposition as the restricted decomposition of U with respect to V, denoted by $RGD_V(U)$.

Let u_a, u_b, v_d, v_c be four points on ∂P appearing in that order in a counterclockwise traversal of ∂P. $P[u_a, u_b; v_c, v_d]$ denotes the region of P obtained by joining the counterclockwise chain of ∂P from u_a to u_b and the clockwise chain of ∂P from v_c to v_d with $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ (see Figure 3.5). We say that a polygonal region $R \subset P$ is *geodesically convex* if for every pair of points $x, y \in R$, we have that $\pi(x, y) \in R$.

Lemma 3.3.2 *[79] $P[u_a, u_b; v_c, v_d]$ is geodesically convex.*

Lemma 3.3.2 implies that the restricted geodesic decomposition of U with respect to chain V can be done entirely within $P[u_a, u_b; v_c, v_d]$. Below, we outline the algorithm to compute this decomposition.

Algorithm 2: $RGD(P[u_a, u_b; v_c, v_d])$

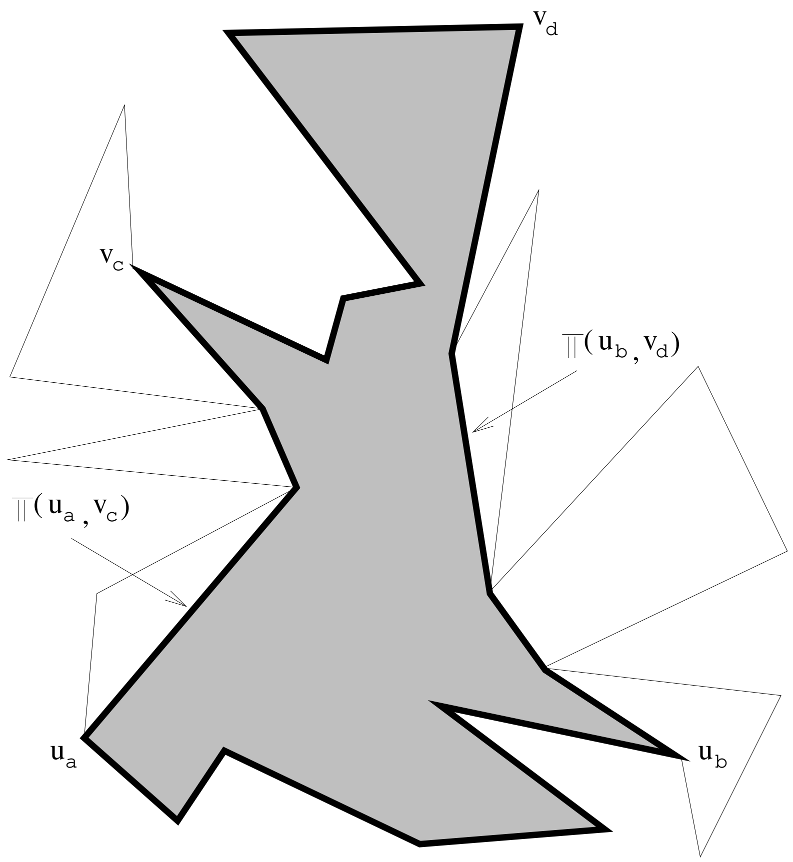

Figure 3.5: $P[u_a, u_b; v_c, v_d]$ is shaded.

1. If $P[u_a, u_b; v_c, v_d]$ is degenerate then

    Find a $v_m \in [v_c, \ldots, v_d]$ such that $d_G(u_a, v_m) = \max\{d_G(u_a, v_j) | c \leq j \leq d\}$;

    Find the point $u_{\min} \in [u_a, \ldots, u_b]$ such that $d_G(u_{\min}, v_m)$
    $= \min\{d_G(u_i, v_m) | a \leq i \leq b\}$.

    Output $u_{\min}$, v_m, $[u_a, \ldots, u_b]$, $d_G(u_{\min}, v_m)$.

2. Else if $(b - a) \leq 2$ and u_a, u_b are vertices then

    Determine $\phi(u_a)$ and $\phi(u_b)$.

    If $\phi(u_a) = \phi(u_b)$ then

      let $v_m = \phi(u_a)$.

      Find point $u_{\min}$ on $[u_a, u_b]$ such that $d_G(u_{\min}, v_m)$
      $= \min\{d_G(p, v_m) | p \in [u_a, u_b]\}$.

      Output $u_{\min}$, v_m, $[u_a, u_b], d_G(u_{\min}, v_m)$.

    Else

      Compute partition points $x_1, x_2, \ldots, x_s$ on edge $[u_a u_b]$.

      Sort these partition points including the endpoints.

      Let $[u_1, u_2, \ldots, u_{s+2}]$ be the points ordered on edge $[u_a, u_b]$.

let $k = \lceil (s+2)/2 \rceil$.
Find a $v_m \in [v_c, \ldots, v_d]$ such that $d_G(u_k, v_m)$
$= \max\{d_G(u_k, v_j) | c \leq j \leq d\}$
Construct and triangulate $P[u_a, u_k; v_m, v_d]$ and $P[u_k, u_b; v_c, m]$;
Call $RGD(P[u_a, u_k; v_m, v_d])$ and $RGD(P[u_k, u_b; v_c, v_m])$.

3. Else if $(b - a) \leq 2$ and u_a, u_b are not both vertices then
 Determine $\phi(u_a)$ and $\phi(u_b)$.
 If $\phi(u_a) = \phi(u_b)$ then
  let $v_m = \phi(u_a)$.
  Find point $u_{\min}$ on $[u_a, u_b]$ such that $d_G(u_{\min}, v_m)$
  $= \min\{d_G(p, v_m) | p \in [u_a, u_b]\}$.
  Output $u_{\min}$, v_m, $[u_a, u_b], d_G(u_{\min}, v_m)$.
 Else
  solve directly by computing upper envelopes.
4. Else
 let $k = \lceil (a+b)/2 \rceil$
 Find a $v_m \in [v_c, \ldots, v_d]$ such that $d_G(u_k, v_m) = \max\{d_G(u_k, v_j) | c \leq j \leq d\}$
 Construct and triangulate $P[u_a, u_k; v_m, v_d]$ and $P[u_k, u_b; v_c, v_m]$;
 Call $RGD(P[u_a, u_k; v_m, v_d])$ and $RGD(P[u_k, u_b; v_c, v_m])$.

Not only does algorithm *RGD* compute the restricted geodesic decomposition of U into polygonal chains $(c_1', c_2', \ldots, c_s')$ such that $\bigcup_{i=1}^{s} c_i' = U$ and for every $x, y \in c_i'$, $\phi_V(x) = \phi_V(y)$, but for each chain c_i' it computes the point on the chain whose distance to its furthest neighbor is minimum. We prove the correctness of the algorithm and at the same time, elaborate on some of the steps such as how to compute the partition in Step 2.

We say that $P[u_a, u_b; v_c, v_d]$ is *degenerate* if $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ are not disjoint. Given a degenerate instance of $P[u_a, u_b; v_c, v_d]$ computing the decomposition of U with respect to V is straightforward. Let x be a point on $\pi(u_a, v_c) \cap \pi(u_b, v_d)$. Every shortest path between a point y in U and a point z in V must contain x. Therefore, the point v_f of V furthest from x is also the point furthest from all points in U. The point v_f can be computed by traversing the shortest path tree of x. Since this tree can be computed in linear time [40], we conclude with the following.

Lemma 3.3.3 *If $P[u_a, u_b; v_c, v_d]$ is degenerate and v_f is the furthest point from $x \in \pi(u_a, v_c) \cap \pi(u_b, v_d)$, then the point v_f on V is $\phi_V(z)$ for all $z \in U$. The point v_f can be computed in time proportional to the size of $P[u_a, u_b; v_c, v_d]$.*

Given an instance of $P[u_a, u_b; v_c, v_d]$, if $(u_a, \ldots, u_b)$ is a polygonal chain, then by the crossing property, we can divide the chain in half and recurse. However, if u_a and u_b are the endpoints of an edge, it is not clear how to proceed. In such a situation, we resolve the problem by partitioning the edge into subedges. We require the following property on each subedge s_i of $[u_a, u_b]$. For every pair of points x, y on s_i, we want the shortest path from x to every vertex v_j on $(v_c, \ldots, v_d)$ to be identical to the shortest path from y to v_j except for the first link. We refer to this property as the *path-invariant* property of a subedge.

To see how to compute a partition of the edge respecting the path-invariant property, let us look at Figure 3.6. In Figure 3.6(a), notice that once the shortest paths from v_i to u_1, and v_i to

u_8 are computed, the path-invariant partition of the edge falls out by extending the edges in both paths to $[u_1, u_8]$. In this partition of the edge, every point on a subedge (u_i, u_{i+1}) has the same shortest path to v_i except for the first link. To extend this partition to two vertices, see Figure 3.6(b). In Figure 3.6(b), the partition with respect to v_i and v_{i+1} differs by only one point located between u_2 and u_3. Again, each subedge has the property that for every point on the subedge, the shortest paths to both v_i and v_{i+1} are the same except for the first link. By continuing in this manner, the edge can be partitioned with respect to the chain $(v_c, \ldots, v_d)$.

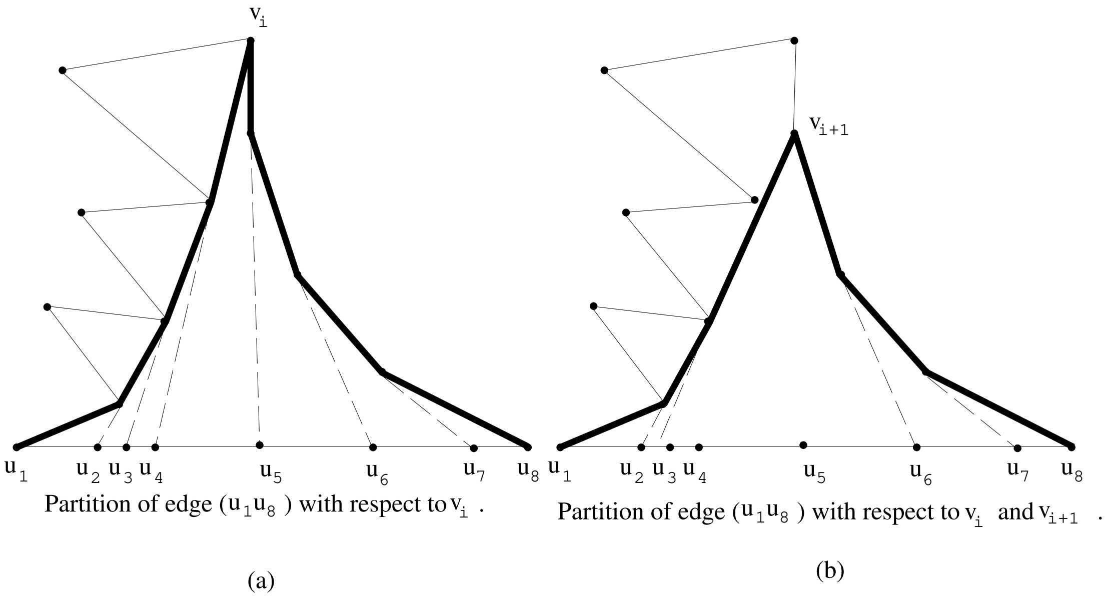

Figure 3.6: Partitioning an edge.

Let m be the size of $P[u_a, u_b; v_c, v_d]$. Since $P[u_a, u_b; v_c, v_d]$ is triangulated, the shortest path tree of u_a and u_b can be obtained in $O(m)$ time using the algorithm of [40]. Once the shortest path trees have been computed, all the partition points on the edge can also be obtained in O(m) time, by traversing the two trees. Finally, $O(m \log m)$ time is used to sort the partition points. Hence, we conclude with the following.

Lemma 3.3.4 *Given an instance of $P[u_a, u_b; v_c, v_d]$ of size m, where $[u_a, u_b]$ is an edge, we can partition in $O(m \log m)$ time the edge $[u_a, u_b]$ into subedges such that each subedge respects the path-invariant property with respect to the chain $(v_c, \ldots, v_d)$.*

The reason we partition the edge into subedges, when faced with an instance of $P[u_a, u_b; v_c, v_d]$ where $[u_a, u_b]$ is an edge, is quite simple. First, it allows us to continue the divide-and-conquer algorithm. Second, the base problem that we are faced with at the end of the recursion can be solved directly because of the path-invariant property. As the algorithm computes the decomposition of the U chain, eventually in Step 3, we are faced with an instance $P[u_a, u_b; v_c, v_d]$ where u_a, u_b are the endpoints of a subedge respecting the path-invariant property, and $(v_c, \ldots, v_d)$ is a polygonal chain. Because of the path-invariant property, we know that the distance from a vertex $v_i \in (v_c, \ldots, v_d)$ to a point x on $[u_a, u_b]$ has the form $k_1 + \sqrt{k^2 + z^2}$ where k_1 is a constant whose value is the geodesic distance from v_i to the last vertex, say v_l, before x on $\pi(v_i, x)$, and $\sqrt{k^2 + z^2}$ is the distance from

v_l to x with k as the orthogonal distance between the line L containing $[u_a, u_b]$ and v_l, and z represents the distance between x and the point on L that is the orthogonal projection of v_l onto L. Consider the example in Figure 3.7. The constant k_1 accounts for the distance from v_i to v_{i+2}. By the path-invariant property, this value is the same for all points on the subedge. The distance from v_{i+2} to x is accounted for by $\sqrt{k^2 + z^2}$.

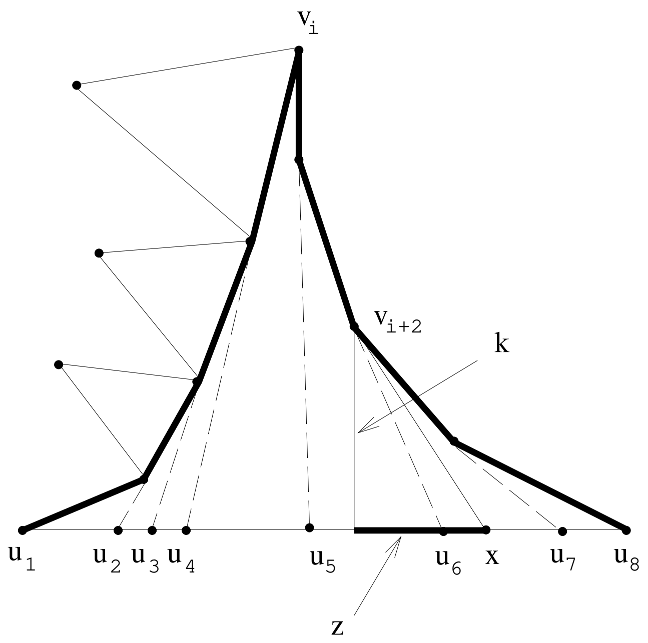

Figure 3.7: Distance function from subedge to vertex.

Let $d_{v_i}(x)$ denote the distance function from $v_i \in (v_c, \ldots, v_d)$ to a point x in $[u_a, u_b]$. These functions are simple and can be used to solve directly the decomposition of $[u_a, u_b]$ into subedges such that for each point in the subedge, the furthest neighbor is the same vertex of $(v_c, \ldots, v_d)$. This can be achieved by computing the upper envelope of the functions $d_{v_i}(x)$ for all $v_i \in (v_c, \ldots, v_d)$. The following lemma gives the key to solving this in time that is linear in the size of the problem instance.

Lemma 3.3.5 *Let p_1, p_2, p_3, p_4 be four points in this order on the boundary of P. If $d_G(p_1, p_4) > d_G(p_1, p_3)$ then $d_G(p_2, p_4) > d_G(p_2, p_3)$.*

Proof: Suppose $d_G(p_1, p_4) > d_G(p_1, p_3)$ and $d_G(p_2, p_3) \geq d_G(p_2, p_4)$. We see that $d_G(p_1, p_4) + d_G(p_2, p_3) > d_G(p_1, p_3) + d_G(p_2, p_4)$. By the relative positions of the points, $\pi(p_1, p_3)$ must intersect $\pi(p_2, p_4)$. Let x be a point on this intersection.

By the triangle inequality, $d_G(p_1, p_4) \leq d_G(p_1, x) + d_G(x, p_4)$, and $d_G(p_2, p_3) \leq d_G(p_2, x) + d_G(x, p_3)$. But since $x \in \pi(p_1, p_3) \cap \pi(p_2, p_4)$, we contradict our assumption, proving the lemma. ■

The above lemma implies that we can compute the upper envelope in linear time simply by inserting the functions in the order $(v_c, \ldots, v_d)$ or the reverse order. Both arguments are symmetric. Let us look at an example to see why this is so. Suppose we are inserting the functions in the order $d_{v_d}(x), d_{v_{d-1}}(x), \ldots, d_{v_c}(x)$. Consider the example in Figure 3.8 where the first three functions have been inserted. The upper envelope consists of $d_{v_d}(x)$ between u_a and u_1, $d_{v_{d-1}}(x)$ between u_1 and u_2, and $d_{v_{d-2}}(x)$ between u_2 and u_b. The next function to be added is $d_{v_{d-3}}(x)$. If $d_{v_{d-3}}(x)$ is below $d_{v_{d-2}}(x)$ between u_2 and u_b then it cannot lie on the upper envelope because if it did, we would have a situation contradicting Lemma 3.3.5. If $d_{v_{d-3}}(x)$ intersects $d_{v_{d-2}}(x)$ between u_2 and u_b then we update the upper envelope by adding the intersection point, but we no longer need to compare $d_{v_{d-3}}(x)$ with any other function on the upper envelope by Lemma 3.3.5. Finally, if $d_{v_{d-3}}(x)$ is above $d_{v_{d-2}}(x)$ between u_2 and u_b then remove the intersection point u_2, remove $d_{v_{d-2}}(x)$ from the upper envelope, and repeat the test on the next piece of the upper envelope, namely $d_{v_{d-1}}(x)$. Therefore, when we add the functions in the order $d_{v_d}(x), d_{v_{d-1}}(x), \ldots, d_{v_c}(x)$, the amount of time spent adding a function can be determined in constant time plus the time proportional to the number of functions and intersection points deleted which is linear time overall. We conclude with the following lemma.

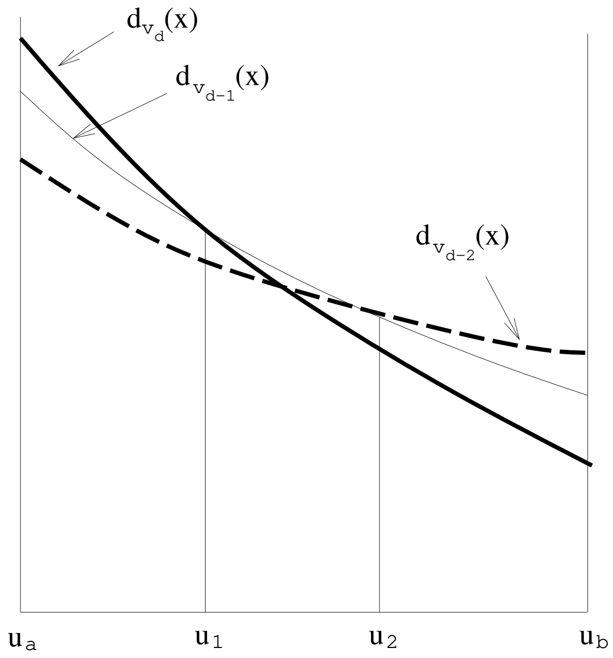

Figure 3.8: Computing upper envelopes.

Lemma 3.3.6 *Given an instance of $P[u_a, u_b; v_c, v_d]$ where $[u_a, u_b]$ is a segment with the path-invariant property, $[u_a, u_b]$ can be decomposed, in time proportional to the size of $P[u_a, u_b; v_c, v_d]$, into subedges such that for each point in the subedge, the furthest neighbor is the same vertex of $(v_c, \ldots, v_d)$.*

The algorithm to compute $RGD_V(U)$ stems from the crossing property described in Lemma 3.3.1. We show that this property holds at all levels of recursion. The algorithm is initiated with a call to $RGD(P[u_a, u_b; v_c, v_d])$. At each invocation, the algorithm either makes two recursive calls with smaller problem instances or solves the problem directly. The calling relation forms a binary tree, which we refer to as the *recursion tree*. A node of this tree having two children is an instance of RGD where two recursive calls were made. A leaf of the recursion tree is an instance of the problem that is solved directly. The root of the tree represents the initial call. The depth of a node in the tree represents its level of recursion.

Lemma 3.3.7 *Let* $U = (u_a, \ldots, u_b)$ *and* $V = (v_c, \ldots, v_d)$. *Given an initial call of* $RGD(P[u_a, u_b; v_c, v_d])$, *every recursive call* $RGD(P[u_q, u_r; v_s, v_t])$ *has the property that for all* $x \in (u_q, \ldots, u_r)$, *we have that* $\phi_V(x) \in (v_s, \ldots, v_t)$.

Proof: We proceed by induction. The initial call has the property that for every $x \in U$, $\phi_V(x)$ is in V. Let us assume, by induction, that all subproblems at depth k in the recursion tree have the desired property. We show that all problems at depth $k+1$ have the desired property given that the property holds at depth k.

There are only two places in the algorithm where a recursive call takes place. Let us first look at the call in Step 4. The same argument holds for the other call in Step 2. Let $P[u_q, u_r; v_s, v_t]$ be an instance of a problem at depth k. By induction, we know that for all $x \in (u_q, \ldots, u_r)$, we have that $\phi_V(x) \in (v_s, \ldots, v_t)$. In Step 4, $P[u_q, u_r; v_s, v_t]$ is split into two instances, $P[u_q, u_k; v_m, v_p]$ and $P[u_k, u_r; v_s, v_m]$. By the crossing property, we know that for all $x \in (u_q, \ldots, u_k)$, we have $\phi_V(x) \in (v_m, \ldots, v_p)$ and for all $x \in (u_k, \ldots, u_r)$, $\phi_V(x) \in (v_s, \ldots, v_m)$. Thus, the lemma follows by induction. ∎

We are now in a position to prove the correctness of algorithm *RGD*.

Theorem 3.3.1 *Algorithm RGD correctly computes the restricted geodesic decomposition of chain* U *with respect to* V.

Proof: By Lemma 3.3.7, if the root of the recursion tree is an instance of $RGD(P[u_a, u_b; v_c, v_d])$ with the property that for all $x \in (u_a, \ldots, u_b)$, we have that $\phi_V(x) \in (v_c, \ldots, v_d)$, then all recursive calls, i.e. all other nodes of the tree, have this property. Therefore, the correctness of the restricted geodesic decomposition of U with respect to V rests on the correctness of the leaves of the recursion tree, that is, the instances of *RGD* that solve the problem directly.

If the leaf instance is degenerate, then the problem is solved directly in Step 1. The correctness of this step is proved in Lemma 3.3.3. If the problem is solved directly in Step 2 (in the first *if* statement), then the correctness is verified by the crossing property. Similarly, if the problem is solved directly in the first part of Step 3, the correctness is guaranteed by the crossing property. Finally, if the problem is solved directly by computing upper envelopes in Step 3, then it is correct by Lemma 3.3.6. Since, we have shown that all instances where the problem is solved directly are correct, the theorem follows. ∎

We now turn our attention to the complexity analysis of algorithm *RGD*. We show that the algorithm runs in $O(n \log n)$ time and uses $O(n)$ space. To do this, we first show that there are $O(\log n)$ levels of recursion. Then we show that an instance of $RGD(P[u_a, u_b; v_c, v_d])$ (excluding recursive calls and sorting of partition points) runs in time proportional to the size of $P[u_a, u_b; v_c, v_d]$. Finally, we show that the total size of all polygons at a particular level of recursion is $O(n)$. The main ideas in the complexity analysis to follow stem from the analysis given in Suri[79].

Lemma 3.3.8 *Algorithm* $RGD(P[u_a, u_b; v_c, v_d])$ *runs in time proportional to the size of* $P[u_a, u_b; v_c, v_d]$, *excluding recursive calls and sorting partition points.*

Proof: Let m be the size of $P[u_a, u_b; v_c, v_d]$. Step 1 runs in time $O(m)$, by Lemma 3.3.3. A furthest neighbor of a point in $P[u_a, u_b; v_c, v_d]$ can be found in $O(m)$ time using the algorithm of [40]. Therefore, the first part of Step 2 runs in $O(m)$ time. Because of the structure of $P[u_a, u_b; v_c, v_d]$, constructing and triangulating the two subpolygons in the second part of Step 2 (in the *Else* statement) and in Step 4 can be done in $O(m)$ by a simple algorithm in [79] or a more complex algorithm of Chazelle [19]. Since we are excluding the sorting of partition points, Step 2 and Step 4 can be done in $O(m)$. Finally, Step 3 can be achieved in $O(m)$ time as proved in Lemma 3.3.6. The lemma follows. ∎

We first show that the number of *distinct* edges among all the polygons constructed by algorithm *RGD* is $O(n)$, where n is the size of the polygon in the initial invocation. Recall that $P[u_a, u_b; v_c, v_d]$ denotes the region of P obtained by joining the counterclockwise chain of ∂P from u_a to u_b and the clockwise chain of ∂P from v_c to v_d with $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ (see Figure 3.5). We refer to $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ as *connecting paths.* There are three types of edges in $P[u_a, u_b; v_c, v_d]$. An edge that belongs to ∂P is a *primary edge*, an edge that is a subedge of an edge belonging to ∂P is a *partition edge*, and an edge that belongs to a connecting path is a *connecting edge.* The number of distinct primary edges is $O(n)$ since all primary edges are contained in the initial polygon. We now show that there are $O(n)$ distinct partition edges and connecting edges.

Lemma 3.3.9 *Let B be a simple polygon. Let a, c be two arbitrary but fixed points on the boundary of B and let b, d be two other points on the boundary of B such that a, b, c, d appear in this order in a counterclockwise traversal of the boundary of B. Then, all edges of $\pi(b, d)$, except perhaps three, belong to $E(a) \cup E(c)$, where $E(\alpha)$ denotes the set of edges in the shortest path tree of B from the point α.*

Proof: The proof is identical to the proof of Lemma 4 in Suri [79], except there are three edges rather than one that do not belong to $E(a) \cup E(c)$ since we consider points on the boundary of the polygon whereas Suri was dealing with vertices of the polygon. ∎

Lemma 3.3.10 *The total number of partition points added is $O(n)$ where n is the size of the initial instance of $RGD(P[u_a, u_b; v_c, v_d])$.*

Proof: Let $P[u_a, u_b; v_c, v_d]$ represent the initial n vertex polygon. We have $U = (u_a, \ldots, u_b)$ and $V = (v_c, \ldots, v_d)$. Let n_u represent the number of vertices in the U chain and n_v the number in the V chain. We refer to an instance of $RGD(P[u_q, u_r; v_s, v_t])$ where $[u_q, u_r]$ is an edge of P and $(v_s, \ldots, v_t)$ is a polygonal chain belonging to ∂P as a *partition instance.*

Notice that a vertex in the V chain can appear in only two partition instances, since each time during the execution of *RGD* that the V chain is divided, only the dividing vertex appears in common in the two ensuing subinstances. Therefore, we can conclude that at most $2n_u$ vertices from the V chain are considered among all partition instances.

Partition points are created by extending the edges in the shortest path between a vertex v_i in the V chain and a vertex u_j in the U chain. So the number of partition points introduced is bounded by the number of distinct edges of all the shortest paths considered to create the partition points. By Lemma 3.3.9, all but three edges of $\pi(v_i, u_j)$ appear in $E(u_a) \cup E(u_b)$. The size of $E(u_a) \cup E(u_b)$ is $O(n)$. Since at most $2n_u$ vertices of the V chain are considered, there are at most $3 \cdot 2n_u \in O(n)$ edges not accounted for by $E(u_a) \cup E(u_b)$. Therefore, a total of $O(n)$ partition points are introduced. ∎

Since a total of $O(n)$ partition points are added given that the size of the initial instance of $RGD(P[u_a, u_b; v_c, v_d])$ is n, we conclude that the total time spent sorting all partition points is $O(n \log n)$. Therefore we have the following.

Lemma 3.3.11 *Given an initial instance of* $RGD(P[u_a, u_b; v_c, v_d])$ *of size* n*, the total time spent sorting partition points is* $O(n \log n)$.

Lemma 3.3.12 *There are* $O(\log n)$ *levels of recursion where* n *is the size of the initial instance of* $RGD(P[u_a, u_b; v_c, v_d])$.

Proof: Let $\Pi = (i_1, i_2, \ldots, i_m)$ represent the the longest root to leaf path in the recursion tree. Each i_k of the path Π represents the problem instance occurring at recursion level k along the path. On any root to leaf path, there can be only one partition instance. Let us suppose that Π has a partition instance and let i_p represent it. The argument is similar if Π does not have a partition instance.

From i_1 to i_p, at each step, the U chain is divided in half as seen in Step 4 of algorithm *RGD*. Therefore, there are $O(\log n)$ instances from i_1 to the partition instance. At the partition instance i_p, by Lemma 3.3.10, at most $O(n)$ points are introduced. Again, from i_p to i_m, at each step the partitioned edge is divided in half as seen in Step 2. So, the length of the path from i_p to i_m is also $O(\log n)$. Therefore, Π has length $O(\log n)$. Since the longest root to leaf path in the recursion tree has length $O(\log n)$, there are $O(\log n)$ levels of recursion. ■

Lemma 3.3.13 *There are* $O(n)$ *distinct edges among all polygons constructed by* $RGD(P[u_a, u_b; v_c, v_d])$.

Proof: By Lemma 3.3.12, the height of the recursion tree is $O(\log n)$ where n is the size of $P[u_a, u_b; v_c, v_d]$. Since the recursion tree is a binary tree, there are $O(n)$ nodes in the tree. This means that at most $O(n)$ polygons are constructed in total. Since each polygon has two connecting paths, at most $O(n)$ connecting paths are constructed in total.

Now, a connecting path joins a point u_i on the U chain to a point v_j on the V chain. By Lemma 3.3.9, all but three edges of $\pi(u_i, v_j)$ appear in $E(u_a) \cup E(u_b)$. The size of $E(u_a) \cup E(u_b)$ is $O(n)$. Since at most $O(n)$ connecting paths are constructed, there are at most $O(n)$ edges not accounted for by $E(u_a) \cup E(u_b)$. This adds up to a total of $O(n)$ distinct connecting edges. By Lemma 3.3.10, there are only $O(n)$ distinct partition edges. By definition, there are only $O(n)$ distinct primary edges. The lemma follows. ■

All that remains to be shown is that the summed complexity of all the polygons constructed in one particular level of recursion is $O(n)$. To do this, we show that a distinct edge can belong to only a constant number of polygons in a particular level of recursion. By the construction of polygons in Step 2 and Step 4, we see that partition edges and primary edges cannot occur in two polygons at the same level of recursion. This follows from the way the U chain and V chain are divided. We now show that a connecting edge can only occur in a constant number of polygons on the same level of recursion.

In order to show this, we must consider the connecting edges as directed. All connecting paths are directed from the U chain to the V chain. Therefore, the edges of the connecting paths are arcs that are directed from one chain to the other. Consider the two paths in Figure 3.9. Both are connecting paths from the U chain to the V chain, and both have the edge e in common. However, e is directed one way in one of the paths and the opposite way in the other. This distinction is important in the analysis to follow.

Lemma 3.3.14 *[79] Let* $a_1, a_2, a_3, b_1, b_2, b_3$ *be six points in this order in a counterclockwise traversal of* P*. Suppose that the directed shortest paths* $\pi(a_1, b_1)$ *and* $\pi(a_3, b_3)$ *have a directed edge* e *in common. Then, the same directed edge* e *also is included in the shortest path* $\pi(a_2, b_2)$.

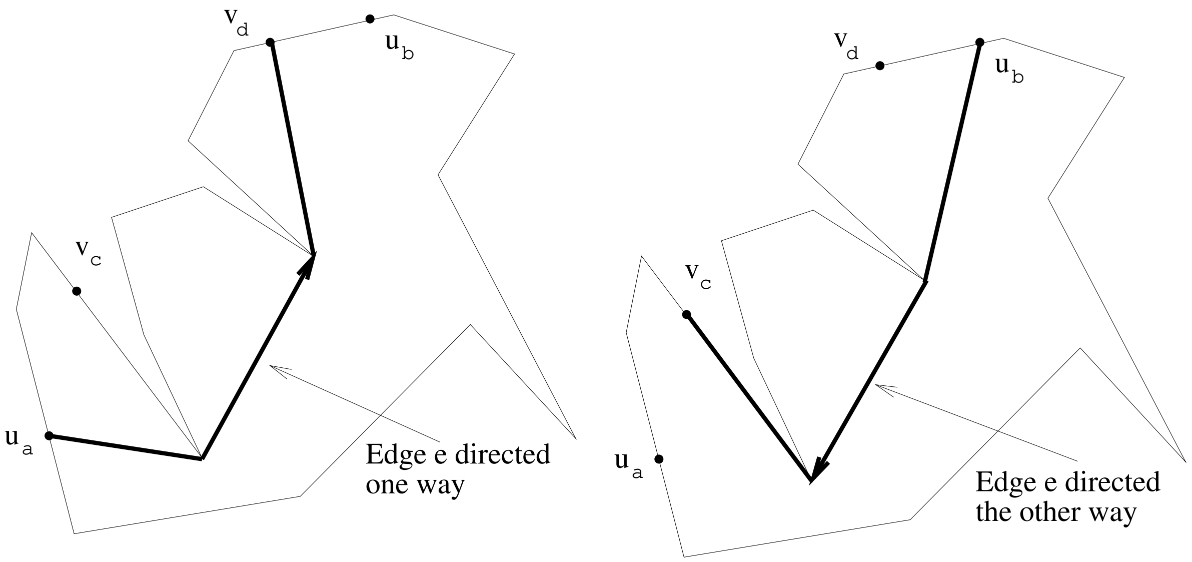

Figure 3.9: Directed edges must be considered.

We only need to consider non-degenerate polygons at the same level of recursion. Given a degenerate polygon at recursion level i, algorithm *RGD* solves the problem directly at this stage. Therefore, since the degenerate polygon is derived from a non-degenerate polygon at level $i-1$, the complexity of the degenerate polygon can be accounted for by the non-degenerate 'parent'.

Lemma 3.3.15 *Let $P[u_{a_1}, u_{b_1}; v_{c_1}, v_{d_1}], P[u_{a_2}, u_{b_2}; v_{c_2}, v_{d_2}]$, and $P[u_{a_3}, u_{b_3}; v_{c_3}, v_{d_3}]$ be three non-degenerate polygons that occur at the same level of recursion, such that $a_1 \leq b_1 \leq a_2 \leq b_2 \leq a_3 \leq b_3$, and $c_3 \leq d_3 \leq c_2 \leq d_2 \leq c_1 \leq d_1$. Then, the directed connecting paths of $P[u_{a_1}, u_{b_1}; v_{c_1}, v_{d_1}]$ and $P[u_{a_3}, u_{b_3}; v_{c_3}, v_{d_3}]$ are edge-disjoint.*

Proof: See Figure 3.10. The proof of this lemma is similar to the proof of Lemma 7 in [79]. Suppose that the two directed connecting paths $\pi(x_1, y_1)$ and $\pi(x_3, y_3)$ share an edge e, where $x_1 \in \{u_{a_1}, u_{b_1}\}$ and $x_3 \in \{u_{a_3}, u_{b_3}\}$. Then by Lemma 3.3.14, $\pi(u_{a_2}, v_{c_2})$ and $\pi(u_{b_2}, v_{d_2})$ must also share edge e, contradicting the fact $P[u_{a_2}, u_{b_2}; v_{c_2}, v_{d_2}]$ is not degenerate. ■

Theorem 3.3.2 *Algorithm RGD computes the restricted geodesic decomposition of chain U with respect to V using $O(n \log n)$ time and $O(n)$ space given an input of size n.*

Proof: The correctness of the algorithm is shown in Theorem 3.3.1. Let $P[u_a, u_b; v_c, v_d]$ be the input polygon of size n to algorithm *RGD*. By Lemma 3.3.8, we know that excluding recursive calls and sorting partition points, algorithm *RGD*$(P[u_q, u_r; v_s, v_t])$ runs in time proportional to the size of $P[u_q, u_r; v_s, v_t]$.

The size of all the polygons constructed at the same level of recursion is $O(n)$ by Lemma 3.3.13 and Lemma 3.3.15. There are $O(\log n)$ levels of recursion. Hence, the total time spent, excluding sorting partition points, is $O(n \log n)$ By Lemma 3.3.11, the time to sort all partitions points is $O(n \log n)$. The theorem follows. ■

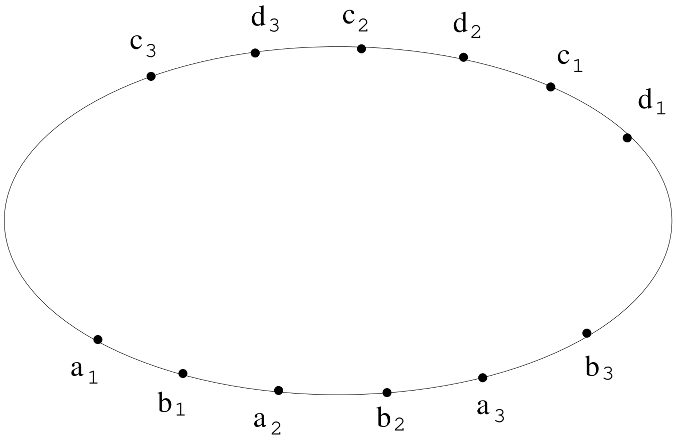

Figure 3.10: Illustration for Lemma 3.3.15

In the next section, we show how to use the restricted geodesic decomposition to solve our initial problem of computing the geodesic decomposition of the boundary.

3.3.3 Geodesic Center Constrained to the Boundary

To compute the geodesic decomposition of the boundary of a simple polygon, we apply the algorithm for restricted decomposition three times. The following lemma of Suri[79] provides the key.

For the following lemma, we assume that $(u_1, u_2, \ldots, u_n)$ is the counterclockwise sequence of vertices of polygon P. We let $(u_a, \ldots, u_b)$ denote the counterclockwise chain of ∂P from u_a to u_b. Let u_i be an arbitrary vertex of P. Let $u_j \in \phi(u_i)$ be a geodesic furthest neighbor of u_i, and $u_k \in \phi(u_j)$ be a geodesic furthest neighbor of u_j. It is possible that $u_i = u_k$. Let us assume, without loss of generality, that u_i, u_j, u_k is the order of these vertices in a counterclockwise traversal of P starting at vertex u_i, then we have the following lemma.

Lemma 3.3.16 *[79] Let u_i be an arbitrary vertex of P. Let $u_j \in \phi(u_i)$ and let $u_k \in \phi(u_j)$, such that u_i, u_j, and u_k are in this order in a counterclockwise traversal of P.*

1. *for any vertex $u_l \in (u_i, \ldots, u_j)$, there exists another vertex $u_m \in (u_j, \ldots, u_i)$ satisfying $u_m \in \phi(u_l)$,*

2. *for any vertex $u_l \in (u_j, \ldots, u_k)$, there exists another vertex $u_m \in (u_k, \ldots, u_i, \ldots, u_j)$ satisfying $u_m \in \phi(u_l)$,*

3. *for any vertex $u_l \in (u_k, \ldots, u_i)$, there exists another vertex $u_m \in (u_i, \ldots, u_j, \ldots, u_k)$ satisfying $u_m \in \phi(u_l)$,*

From the above lemma, we can conclude that to compute the geodesic decomposition of P, we simply solve the following three instances of the restricted geodesic decomposition of P.

Instance 1 The U chain is $(u_i, \ldots, u_j)$ and the V chain is $(u_j, \ldots, u_k, \ldots, u_i)$.

Instance 2 The U chain is $(u_j, \ldots, u_k)$ and the V chain is $(u_k, \ldots, u_i, \ldots, u_j)$.

Instance 3 The U chain is $(u_k, \ldots, u_i)$ and the V chain is $(u_i, \ldots, u_j, \ldots, u_k)$.

Therefore, we have the following theorem.

Theorem 3.3.3 *The geodesic decomposition of a simple polygon can be computed in $O(n \log n)$ time and $O(n)$ space given an input of size n.*

Once the geodesic decomposition of a polygon P has been computed, the boundary-constrained geodesic center can be computed as follows. Let $(c_1, c_2, \ldots, c_t)$ represent the polygonal chains in the geodesic decomposition of the boundary of P where $\bigcup_{i=1}^{t} c_i = P$ and for every $x, y \in c_i$, $\phi(x) = \phi(y)$. For each c_i, compute the point $x \in c_i$, with the property that the geodesic distance from x to $\phi(x)$ is smallest compared to all other points in c_i. In other words, $d_G(x, \phi(x)) = \min_{\forall y \in c_i}\{d_G(y, \phi(y)\}$. The point x is referred to as the *candidate* for the chain c_i. In fact, algorithm *RGD* already computes the candidates for each chain as seen in steps 1, 2 and 3 of the algorithm. We conclude with the following theorem.

Theorem 3.3.4 *The boundary-constrained geodesic center of polygon P is the candidate x^*, such that $d_G(x^*, \phi(x^*)) = \min_{\forall \text{ candidates } y}\{d_G(y, \phi(y)\}$.*

Proof: Suppose that x^* is not the boundary-constrained geodesic center of polygon P. Let z be the boundary-constrained geodesic center of polygon P. Now, z is on some chain c_i of the geodesic decomposition of P. Since it is the boundary-constrained geodesic center of polygon P, it must be the candidate for chain c_i. The geodesic distance from x^* to $\phi(x^*)$ is less than or equal to the geodesic distance from z to $\phi(z)$ by definition. If $d_G(x^*, \phi(x^*)) < d_G(z, \phi(z))$, then z cannot be the boundary-constrained geodesic center. If $d_G(x^*, \phi(x^*)) = d_G(z, \phi(z))$, then x^* is also a boundary-constrained geodesic center. Both are contradictions, thereby proving the theorem. ■

3.3.4 Geodesic Center Constrained to a Polygonal Region

In this section, we address the problem of computing the geodesic center of a simple polygon P constrained to lie inside a simple polygon Q, where Q is contained in P. We denote this center as $GC_Q(P)$. If Q equals P then we simply have the geodesic center of the polygon P. We can further restrict the geodesic center to lie on the boundary of polygon Q, denoted $GC_{\partial Q}(P)$. In this case, if Q equals P, then we have the geodesic center constrained to the boundary of P. The reason we differentiated the problem of computing the geodesic center constrained to the boundary from this problem is that we use the geodesic furthest point Voronoi diagram to solve this problem, but to solve the former problem, we were able to avoid computing the geodesic furthest point Voronoi diagram by modifying Suri's algorithm [79]. The arguments we use to solve this problem are similar to the arguments used to solve the Euclidean center constrained to a polygon region.

Since in this and the following subsection we make extensive use of the *GFPVD*(P), let us review a few of its properties. In order to use the algorithm of [3], we assume that no vertex is geodesically equidistant from two other vertices. This can always be guaranteed by applying a slight perturbation to the vertices if the condition is violated. Like its Euclidean counter-part, *GFPVD*(P) partitions P into cells, $V(p_i)$, such that for every point $p \in V(p_i)$, the point p_i is a furthest geodesic neighbor of p. A vertex of the *GFPVD*(P) is a point that is geodesically equidistant to three vertices furthest from it. An edge between two Voronoi vertices is either a straight edge or a hyperbolic arc. Finally, the boundary of a Voronoi cell consists of a concatenation of straight edges and hyperbolic arcs. For more geometric properties of geodesic furthest point Voronoi diagrams, the reader is referred to [3, 4].

Lemma 3.3.17 *[4, 65] The geodesic center of a simple polygon P lies on the midpoint of the geodesic diameter of P ($GDIAM(P)$) or on a vertex of the $GFPVD(P)$.*

When the geodesic center of the polygon P lies on the midpoint of the geodesic diameter, it has a special property. Let $bis(a,b)$ represent the geodesic bisector of a and b inside P, i.e. for every point x on $bis(a,b)$, $d_G(x,a) = d_G(x,b)$. Let a, b be two points of polygon P, then we have the following.

Lemma 3.3.18 *If the midpoint m of $\pi(a,b)$ lies on the interior of the edge separating cells $V(a)$ and $V(b)$ of $GFPVD(P)$, then m is the geodesic center P and $\pi(a,b)$ is the geodesic diameter of P.*

Proof: We proceed by contradiction. Suppose m is not the geodesic center, and let c be the geodesic center. The bisector $bis(a,b)$ partitions polygon P into two parts. Let P_a represent the part where $\forall x \in P_a, d_G(x,a) < d_G(x,b)$ and P_b be the part where $\forall x \in P_b, d_G(x,b) < d_G(x,a)$. If c is on $bis(a,b)$, then $d_G(c,a) > d_G(m,a)$ since m is on $\pi(a,b)$ and geodesics are unique. If $c \in P_a$ then $d_G(c,b) > d_G(m,b)$ since $\pi(c,b)$ must intersect $bis(a,b)$ at some point x by the Jordan Curve Theorem and $d_G(x,b) \geq d_G(m,b)$. Similarly, if $c \in P_b$ then $d_G(c,a) > d_G(m,a)$. Therefore, by contradiction, m must be the geodesic center. Since m is the geodesic center, it follows that $\pi(a,b)$ is a geodesic diameter. ■

Before continuing, we need a few definitions. Let a, b, c be three points in a simple polygon P. The geodesic angle $\angle abc$ is the smaller of the two angles between the first link on the geodesic path from b to a and the first link on the path from b to c. Now, consider the paths $\pi(a,b), \pi(b,c)$, and $\pi(a,c)$. There exist points a', b', and c' such that the paths $\pi(a,b)$ and $\pi(a,c)$ intersect in the path $\pi(a,a')$, the paths $\pi(b,c)$ and $\pi(b,a)$ intersect in the path $\pi(b,b')$, and the paths $\pi(c,a)$ and $\pi(c,b)$ intersect in the path $\pi(c,c')$. The three paths $\pi(a',b')$, $\pi(b',c')$, and $\pi(c',a')$ form what is known as a *geodesic triangle*, denoted $\triangle a'b'c'$ (see Figure 3.11). The vertices a', b', c', are the only convex vertices of the geodesic triangle and are referred to as the *peaks* of the triangle. Pollack et al.[65] proved the following property concerning geodesic triangles.

Lemma 3.3.19 *[65] If the geodesic angle $\angle ba'c$ at a' is greater than or equal to $\pi/2$, then $d_G(b,c) > d_G(a',b), d_G(a',c)$*

Lemma 3.3.20 *The geodesic center of P constrained to lie in Q is the midpoint m of $GDIAM(P)$ provided that m is the geodesic center of P and lies in Q.*

Proof: Follows from Lemma 3.3.17. ■

To address the problem of determining the location of $GC_Q(P)$ when it does not satisfy the conditions of the above lemma, we establish the following lemmas. Let a, b be two vertices of P such that they each have a corresponding cell $V(a)$ and $V(b)$, respectively, which are adjacent separated by an edge e in $GFPVD(P)$. Also, $\pi(a,b)$ is not the geodesic diameter of P. Let x be a point on the interior of e, and let $\epsilon > 0$ be any small constant.

Lemma 3.3.21 *There exists a point $y \in e$ with $d_G(x,y) < \epsilon$ such that $d_G(y,a) < d_G(x,a)$ and $d_G(y,b) < d_G(x,b)$.*

Proof: The edge e must lie on $bis(a,b)$, since the points on e are equidistant from both a and b. The point x must be a peak of the geodesic triangle formed by the paths $\pi(x,a), \pi(x,b)$,

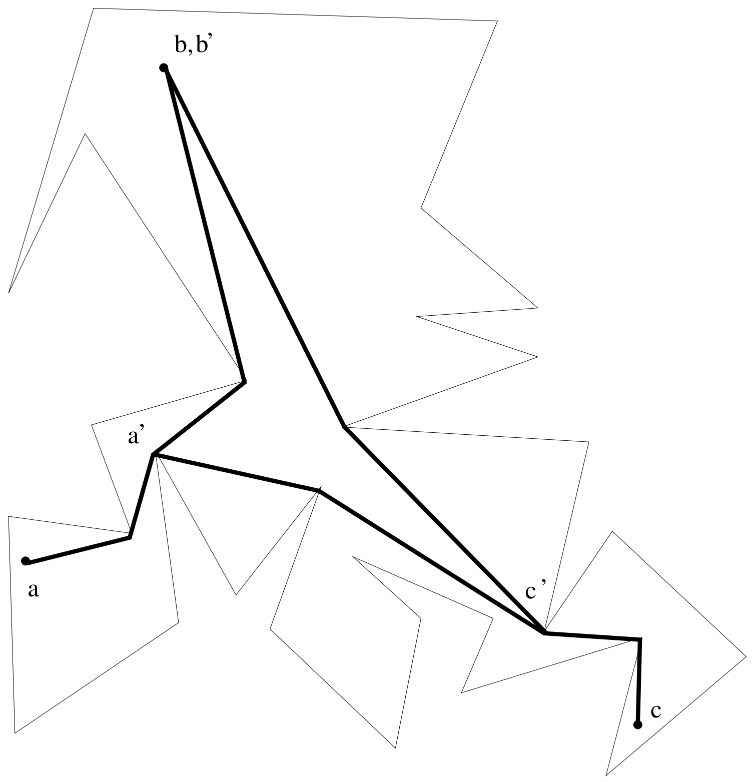

Figure 3.11: A Geodesic Triangle.

and $\pi(a, b)$ since otherwise x would be on the path $\pi(a, b)$ which would imply that $\pi(a, b)$ was a geodesic diameter by Lemma 3.3.18. Also, a portion of e must be contained in the geodesic triangle, since x is on the interior of e. Let y be a point on e in the geodesic triangle. Since the geodesic angle $\angle ayb$ must be no greater than π, by Lemma 3.3.19 we conclude that $d_G(y, a) < d_G(x, a)$ and $d_G(y, b) < d_G(x, b)$. The lemma follows. ■

Lemma 3.3.22 *A point b of P cannot lie in $V(b)$.*

Proof: Let $x \in P$ be a point distinct from b. If $b \in V(b)$ then $d_G(b, b) = 0$. However, $d_G(b, x) > 0$ which contradicts the fact that $b \in V(b)$. ■

We now complete the characterization of $GC_Q(P)$.

Lemma 3.3.23 *If the geodesic center of P constrained to lie in Q is not the midpoint of $GDIAM(P)$, then it lies on one of the following points:*

1. *a vertex of the $GFPVD(P)$ contained in Q,*

2. *a proper intersection point of the $GFPVD(P)$ and the boundary of Q,*

3. *a vertex of the polygon Q,*

4. *a point x on an edge e of Q with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d_G(y, \phi(x)) \geq d_G(x, \phi(x))$.*

Proof: If $GC_Q(P)$ does not lie on any of the points mentioned in the statement of the lemma, then it must lie in one of the regions described in the following four cases. We show that each of these cases leads to a contradiction. For simplicity of exposition, let $c = GC_Q(P)$.

Case 1: c is a point in the interior of a cell of the $GFPVD(P)$, and in $int(Q)$. Let $V(b)$ be the cell containing c. By the Jordon Curve Theorem [64], $\pi(bc)$ must intersect ∂P or $V(b)$ since $b \notin V(b)$ by Lemma 3.3.22. Let x be the intersection point closest to c. The point x must be in $V(b)$. Therefore b is a furthest neighbor of both x and c. However, $d_G(x, b) < d_G(c, b)$ by construction. Hence, we have a contradiction.

Case 2: c is a point in the interior of a cell of the $GFPVD(P)$, and in the interior of an edge e of Q but does not satisfy the property that $\forall y \in e$, if $\phi(y) = \phi(c)$ then $d_G(y, \phi(c)) \geq d_G(c, \phi(c))$. Since the latter property is not satisfied, a point $x \in e$ such that $\phi(x) = \phi(c)$ and $d_G(x, \phi(c)) < d_G(c, \phi(c))$ must exist. However, the very existence of x contradicts that c is the geodesic center of P constrained to lie in Q.

Case 3: c is a point in the interior of an edge e of the $GFPVD(P)$, and in $int(Q)$. Let $V(a)$ and $V(b)$ be the two cells separated by the edge e. Since c is not the midpoint of the geodesic diameter of P, by Lemma 3.3.21 we know that there exists a point x in e and in $int(P)$ such that $d_G(x, a) < d_G(c, a)$ and $d_G(x, b) < d_G(c, b)$. This contradicts that c is $GC_Q(P)$.

Case 4: c is a point in the interior of an edge e_v of the $FPVD(S)$, and in the interior of an edge e_p of P such that e_v and e_p intersect but not properly. Same argument as Case 3.

∎

We outline the algorithm to compute $GC_Q(P)$.

Algorithm 3: *Geodesic Center of P constrained to lie in Q*

Input: A simple polygon $P = \{p_1, p_2, \ldots, p_n\}$, and a simple polygon $Q = \{q_1, q_2, \ldots, q_n\}$ with $Q \subset P$.

Output: $GC_Q(P)$

1. Compute the $GFPVD(P)$ using the algorithm of Aronov et al.[3].
2. Compute $GC(P)$ using the algorithm of Pollack et al.[65].
3. Preprocess Q in $O(n \log n)$ time for point inclusion testing in $O(\log n)$ time using the algorithms of Kirkpatrick [45] or Sarnak and Tarjan [75].
4. If $GC(P)$ is contained in Q then exit with $GC(P)$ as $GC_Q(P)$.
5. Preprocess P for shortest path queries using the algorithm of Guibas and Hershberger [39].
6. Compute the set of vertices of $GFPVD(P)$ contained in Q. Let V_c represent this set.
7. Compute the set of intersections $I_c = \{i_1, i_2, \ldots, i_k\}$ of Q with $GFPVD(P)$ using the algorithm of Chan[18].
8. Partition each edge e_i of Q such that for every pair of points $x, y \in e_i$, we have that $\phi(x) = \phi(y)$. Denote the j^{th} partition of e_i by e_{ij}.

9. For each e_{ij}, compute the point on e_{ij} closest to $\phi(e_{ij})$. If this point is not an endpoint of e_{ij}, place it in the set E_c.
10. Let P_c represent the vertices of Q.
11. Let $CAN = V_c \cup I_c \cup P_c \cup E_c$.
12. Find the set of points $G = \{x \in CAN \mid d_G(x, \phi(x)) = \min_{y \in CAN} d_G(y, \phi(y))\}$
13. Output the set G.

Notice that we assumed that the number of vertices of Q equals the number of vertices of P. Clearly, this need not be the case, however, this assumption simplifys the complexity of notation. It is quite straightforward to repeat the complexity analysis when P and Q have different cardinalities.

Theorem 3.3.5 *Given a polygon $P = \{p_1, p_2, \ldots, p_n\}$ and a polygon $Q = \{q_1, q_2, \ldots, q_n\}$ contained in P, we can compute the geodesic center of P constrained to lie in Q in time $O(n(n+k))$ where n is the size of the input and k is the number of intersections between the edges of the $GFPVD(P)$ and Q.*

Proof: The correctness of the algorithm follows from Lemmas 3.3.20 and 3.3.23.

Let us analyze the complexity of the algorithm. Step 1 of the algorithm can be computed in $O(n \log n)$ time using the algorithm of Aronov et al.[3]. Step 2 can be computed in $O(n \log n)$ time using the algorithm of Pollack et al.[65]. Preprocessing for point inclusion can be done in $O(n \log n)$ using the algorithm of Kirkpatrick [45] or Sarnak and Tarjan [75]. Step 5 can be achieved in $O(n \log n)$ time by using the algorithm of Guibas and Hershberger[39]. By preprocessing the polygon for shortest path queries, in $O(\log n)$ time the geodesic distance between two points can be recovered and in $O(\log n + m)$ time the geodesic path between two points can be recovered where m is the length of the path. Step 6 can be done in $O(n \log n)$ time using the point inclusion test. Computing the intersections between $GFPVD(P)$, which consists of straight edges and hyperbolic arcs, and Q, which consists only of straight edges, can be computed in $O(n \log n + k)$ time where k is the number of intersections between Q and $GFPVD(P)$ using the algorithm of Chan [18]. Once the intersection points have been computed, Step 8 can be achieved in $O(k \log n)$ time. In Step 9, to compute the point on e_{ij} closest to $\phi(e_{ij})$, we first compute the geodesic path from the endpoints of e_{ij} to $\phi(e_{ij})$ in $O(\log n + m)$ time where m is the length of the two paths using [39]. Once the two paths have been computed, finding the point geodesically closest to $\phi(e_{ij})$ can be done $O(m)$ time in the manner described in Subsection 3.3.2. Note that $O(m) \in O(n)$. Step 9 is executed $O(\max\{k, n\})$ times, thus the complexity is $O(n(n+k))$. Step 12 can be computed in $O(k)$ time. Therefore, the total complexity of the algorithm is $O(n(n+k))$ time. ∎

3.3.5 Geodesic Center Constrained to a Polygonal Chain

With a slight modification, Algorithm 3 can compute the geodesic center of P constrained to lie on the boundary Q, $GC_{\partial Q}(P)$. These modifications are outlined below.

Lemma 3.3.24 *The geodesic center of P constrained to lie on the boundary of Q is the midpoint m of $GDIAM(S)$ provided that m is the geodesic center of P and lies on the boundary of Q.*

Proof: Follows from Lemma 3.3.17. ∎

Lemma 3.3.25 *If the geodesic center of P constrained to lie on the boundary of Q is not the midpoint of $GDIAM(P)$, then it lies on one of the following points:*

1. *a vertex of the $GFPVD(P)$ on the boundary of Q,*
2. *a proper intersection point of the $GFPVD(P)$ and the boundary of Q,*
3. *a vertex of the polygon Q,*
4. *a point x on an edge e of Q with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d_G(y, \phi(x)) \geq d_G(x, \phi(x))$.*

Proof: Similar case analysis as the proof of Lemma 3.3.23. ∎

Lemma 3.3.24 and Lemma 3.3.25 completely characterize the location of $GC_{\partial Q}(P)$. The modifications to Algorithm 3 for computing these points are straightforward. Therefore, we conclude with the following.

Theorem 3.3.6 *Given a polygon $P = \{p_1, p_2, \ldots, p_n\}$ and a polygon $Q = \{q_1, q_2, \ldots, q_n\}$ contained in P, we can compute the geodesic center of P constrained to lie on the boundary of Q in time $O(n(n+k))$ where n is the size of the input and k is the number of intersections between the edges of the $GFPVD(P)$ and Q.*

3.4 Constrained Link Center

In this section, we consider the second property attributed to a good pin gate location. Recall that the second property states that the maximum number of turns that the liquid takes on its path from the pin gate to any point in the object should be small. The link metric provides a geometric interpretation of this property. The link metric measures the number of turns or bends in a path between two points. We need a few definitions about link paths before continuing.

The *link distance* between two points x and y inside a polygon P, denoted $d_L(x, y)$, is the minimum number of edges in any polygonal path connecting x and y without intersecting the boundary of P. A path $\pi_L(x, y)$ between x and y is a *minimum link path* provided that the number of edges in $\pi_L(x, y)$ is equal to $d_L(x, y)$. The *k-neighborhood* or *k-disk* about a point $x \in P$ is defined as $N_k(x) = \{y \in P \mid d_L(x, y) \leq k\}$, and the *covering radius* $c(x)$ of x is the smallest k such that $P \subset N_k(x)$. The *link radius* is defined by $r_L(P) = \min_{x \in P} c(x)$ and the *link center* of P is defined by $LC(P) = \{x \in P \mid c(x) = r_L(P)\}$. In essence, the link center is the set of points in P whose maximum link distance to any point in P is minimized, precisely the set of potential pin gates satisfying the second property of a suitable pin gate.

We review the problem of computing the link center of a simple n vertex polygon P. The problem of computing the link center was first addressed by Lenhart et al.[51] who provided a simple $O(n^2)$ time algorithm to compute $LC(P)$. Note that the link center of P is not necessarily a point as is the case with the geodesic center of P, but the link center may in fact be a geodesically convex region contained in P. Later Djidjev et al.[25] reduced the time complexity of computing $LC(P)$ to $O(n \log n)$. Therefore, to compute the link center of a simple polygon, either of these two algorithms may be used.

The problem of computing the link center constrained to the boundary of a polygon P, denoted as $LC(\partial P)$, has not been addressed. In this section, we provide a simple algorithm to compute the set $LC^*(\partial P)$ which is a subset of $LC(\partial P)$. In some cases $LC^*(\partial P)$ is in fact be equivalent to $LC(\partial P)$.

3.4.1 Link Center Constrained to the Boundary

In this section, we provide a simple algorithm for computing $LC^*(\partial P)$, which is a subset of $LC(\partial P)$. The following very simple observations form the basis of the algorithm.

Observation 3.4.1 *If $LC(P) \cap \partial P$ is non-empty, then $LC(\partial P) = LC(P) \cap \partial P$.*

Observation 3.4.2 *If a point $z \notin LC(P)$ is visible from a point $x \in LC(P)$, then $c(z)$ is one greater than $c(x)$.*

Two points x, y in polygon P are said to be *visible* provided that the line segment $[x, y]$ is in P. Given a set of points X in polygon P, the *strong visibility set* of X in P is $\{z \in P \mid \forall\, x \in X, [xz] \in P\}$ and the *weak visibility set* of X in P is $\{z \in P \mid \exists\, x \in X, [xz] \in P\}$. If X happens to be a simple polygon inside P, Ghosh [38] has shown that the *weak visibility set* of X in P is also a simple polygon, referred to as the *weak visibility polygon* of P from X and denoted by $WVP(X, P)$. We now outline the algorithm.

Algorithm 4: *Compute $LC^*(\partial P)$*

1. Compute $LC(P)$.
2. Let $LC_1 = LC(P) \cap \partial P$. If LC_1 is non-empty, exit with LC_1.
3. Compute the weak visibility polygon of P from $LC(P)$.
4. Let $LC_2 = WVP(LC(P), P) \cap \partial P$. Exit with LC_2.

If LC_1 is non-empty, then $LC(\partial P)$ is equal to LC_1. If on the other hand, LC_1 is empty, then the set LC_2 must be a subset of $LC(\partial P)$ since the link center of the polygon is contained strictly in the interior of P and the covering radius of every point in LC_2 is one greater than the covering radius of a point in the link center. The complexity of the algorithm is dominated by Step 1 which can be computed in $O(n \log n)$ time using the algorithm of [25]. A simple modification to the algorithm in [25] is needed to compute the intersection of $LC(P)$ with the boundary of P in the same time complexity. Step 3 can be performed in $O(n)$ time using the algorithm of [38]. The parts of $WVP(LC(P), P)$ that are part of the boundary of P can be identified during the computation of the weak visibility polygon. Therefore, we conclude with the following theorem.

Theorem 3.4.1 *$LC^*(\partial P)$ can be computed in $O(n \log n)$ time.*

3.5 Discussion

Of the solutions presented in this chapter, computing the Euclidean center, with or without constraints, as well as the link center, with or without constraints, are both conceptually and computationally simpler than computing the geodesic center. However, the Euclidean center may not always be a good candidate for the location of a pin gate as pointed out in Section 3.3. The link center considered alone may also not be a suitable candidate since liquid inside a mold does not necessarily travel along a link path. Combining these two constraints may provide a better approximation (e.g. computing the Euclidean center constrained to lie in the link center).

The geodesic center, although computationally more expensive, seems to be a better measure in terms of the distance the liquid travels inside a mold. A combination of the link and geodesic centers may reap the benefits of both properties of an ideal pin gate location being satisfied. For example, computing the geodesic center constrained to lie in the link center may provide a better solution than considering the geodesic center by itself.

Chapter 4

Gravity Casting in Two Dimensions

4.1 Introduction

'Mold orientation during fill is a cut-and-try process to find the most favorable position.' [71]

The above quote points out one of the key problems in gravity casting: find a favorable orientation for a mold during fill that allows the most complete fill and minimizes the number of surface defects. This problem is difficult when the focus is on the fluid dynamics and physics of the whole molding process. However, when viewed from a purely geometric perspective, finding a favorable mold orientation no longer need be a cut-and-try process. Our motive in this chapter is to study gravity casting from a geometric perspective and present algorithms to find mold orientations that allow the most complete fill for molds modelled as simple polygons. We begin by defining a geometric model of the gravity casting process referred to as the *gravity model*.

4.1.1 Geometric Model

The point on the polygon boundary from which the liquid is poured into the polygon is called the *pin gate*. A *venting hole* is a point from which only air and not any liquid is allowed to escape. The pin gate is considered to be a venting hole. We assume that neither the liquid being poured into the mold, nor the air in the mold are compressible. Finally, we assume that air cannot bubble out through the liquid.

The sole force acting on the liquid is gravity. When a direction of gravity is not specified, we assume, for simplicity of exposition, that it acts in the negative y-direction. Thus, if only one pin gate is used, we assume it to be a point on the boundary with the highest y-coordinate, since otherwise, the polygon cannot be completely filled.

When liquid is poured into a polygon, the level of the liquid rises in the direction opposite that of gravity. We assume that the advancing *front* of the rising liquid is a line. The lowest horizontal line such that all the liquid in the polygon is contained below it, is defined as the *level line*.

It is possible for the level line to be higher than the level of the liquid in some section of the polygonal mold. For example, the situation depicted in Figure 4.2 can occur while the mold is being filled with liquid. Thus we define a *level chord* to be the horizontal chord representing the level of liquid in the subpolygon lying below the chord. The region inside the polygon and above the level line contains air. Similarly, the subpolygon containing the level chord, below the level line inside the polygon, contains air above the level chord.

When the level line contains the pin gate, we say the polygon is *maximally filled*. A region containing air in a maximally filled polygon is called an *air pocket*. The highest point (there may

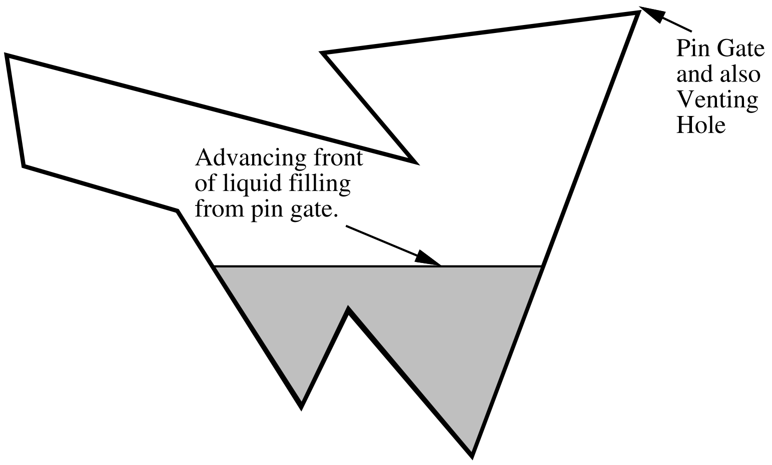

Figure 4.1: Illustration of the gravity model.

be more than one) of an air pocket in a maximally filled mold is the *peak* of the air pocket. A polygon is said to be *k-fillable* if there exists a fixed orientation of the polygon, a placement of the pin gate and a placement of $k-1$ venting holes such that when liquid is poured into the polygon through the pin gate, there are no air pockets when the polygon is maximally filled. A polygon is said to be *k-fillable with re-orientation* provided that the polygon can be re-oriented and filled from a new pin-gate after partial filling from an initial orientation and pin gate. We assume that after the completion of a partial filling, the liquid that is poured into the polygon hardens. The number k in this case refers to the number of times that the polygon needs to be re-oriented before it is completely filled. Notice that both definitions are identical when $k = 1$. Unless stated otherwise, we will always refer to *k-fillable* as filling from a fixed orientation.

4.2 The Decision Problem

The first problem we address is to determine given a simple polygon in a fixed orientation, whether or not the polygon is 1-fillable in that orientation. We present a linear time algorithm to solve this problem. Let g be the point on the unit circle representing the direction of gravity. We make the following key observation (refer to Figure 4.3).

Observation 4.2.1 *The peak of an air pocket is a local maximum of polygon P with respect to the direction of gravity. It is either a convex vertex v_i of the polygon P such that* $\mathrm{ray}(v_iv_{i-1}) \in NH(g)$ *and* $\mathrm{ray}(v_iv_{i+1}) \in NH(g)$ *or a convex edge e_i with endpoints v_i and v_{i+1} such that* $\mathrm{ray}(v_iv_{i-1}) \in NH(g)$ *and* $\mathrm{ray}(v_{i+1}v_{i+2}) \in NH(g)$. *Such a vertex or edge will be referred to as a local maximum vertex or local maximum edge.*

The above observation forms the basis of the following theorem characterizing 1-fillable polygons. Given a point p (or horizontal edge e) in the plane, let $h(p)$ ($h(e)$) denote the horizontal line containing p (e).

Theorem 4.2.1 *A polygon is 1-fillable if and only if it contains one local maximum vertex or one local maximum edge with respect to the direction of gravity.*

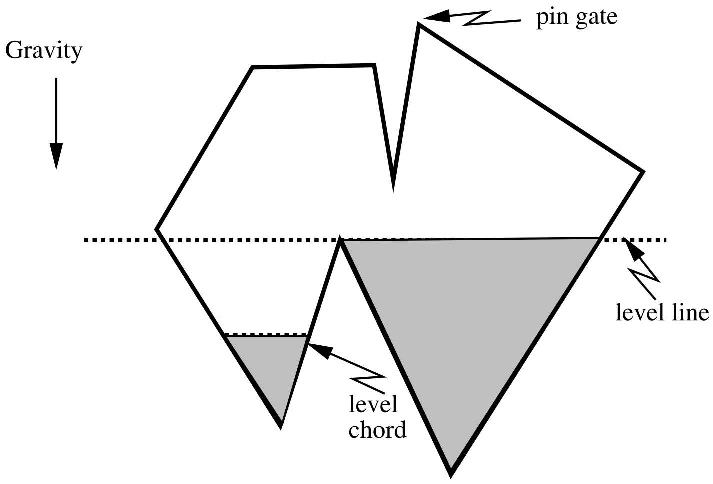

Figure 4.2: Level line and level chord.

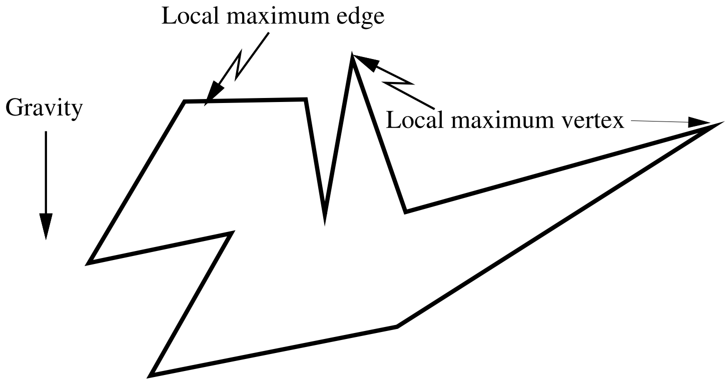

Figure 4.3: Illustration of local maximum vertex and edge.

Proof:

($\Rightarrow$) We first show that a polygon with more than one local maximum vertex will contain at least one air pocket when filled. We proceed by contradiction. Suppose a polygon P containing at least two local maximum vertices can be maximally filled with no air pockets. Let c_1 and c_2 be the two local maxima, with c_1 having the larger y-coordinate.

Since both c_1 and c_2 are both local maxima, the polygonal chain between c_1 and c_2 contains a reflex vertex v or reflex edge e such that the vertices adjacent to v lie above $h(v)$ or the vertices adjacent to e lie above $h(e)$. Let us assume it is a vertex v. A similar argument holds for edge e.

Since the polygon can be maximally filled with no air pockets, at some point in time while liquid is being poured in the polygon, there will exist a level chord $[bc]$ containing v. Let b, v, c be the sequence of these three points when viewed in clockwise order starting at b. Since v is a reflex vertex, both $[vb]$ and $[cv]$ are chords. Let P_1 be the subpolygon consisting of the clockwise chain from b to v and the edge $[vb]$ and let P_2 be the subpolygon consisting of the clockwise chain from

v to c and the edge $[vc]$. Without loss of generality, let the pin gate be contained in P_2. Polygon P_1 must contain some air since $[bc]$ is a level chord.

Now, every path from a point in P_1 to the pin gate must intersect $[vb]$. But this implies that the air in P_1 is trapped since we assumed that air cannot bubble through liquid. Thus, P_1 contains an air pocket, contradicting the fact that P can be filled with no air pockets. A similar argument holds for local maximum edges.

($\Leftarrow$) We now show that a polygon with at least one air pocket when filled must have more than one local maximum vertex or edge. Let P be a filled polygonal mold with an air pocket. Let p be the peak of the air pocket, and pg be the pin gate. By Observation 4.2.1, either p is a local maximum vertex or edge. Also, by assumption, pg is the highest point on the polygon with respect to the direction of gravity. Therefore, it is also a local maximum vertex or contained in a local maximum edge. Since the pin gate cannot be the peak of an air pocket, the polygon contains at least two local maxima. ■

Corollary 4.2.1 *A polygon P is 1-fillable if and only if $\forall p \in P$, the shortest path from p to the pin gate is monotonic with respect to the direction of gravity.*

A chain $C_{ij}(P)$ is *monotonic* with respect to direction Θ if the projections of the vertices $p_i, p_{i+1}, \ldots, p_j$ onto a line $L(\Theta)$ are ordered as the vertices in $C_{ij}(P)$.

A simple linear time algorithm for the decision problem is implied by Theorem 4.2.1. By testing locally, with respect to the direction of gravity, every convex vertex to determine whether or not it is a local maximum vertex, and testing every edge to determine whether or not it is a local maximum edge, we can determine if a polygon is 1-fillable with respect to the direction of gravity. In fact, the number of local maximum vertices and edges determines the number of venting holes that need to be placed in order to fill the polygon in the given orientation.

Lemma 4.2.1 *The number of venting holes needed is equal to the number of local maximum vertices and edges.*

Proof: ($\Rightarrow$) We proceed by induction on the number of local maximum vertices and edges. Recall that gravity is assumed to point in the negative y direction.

Basis: The number of local maximum vertices or edges, max, is 1. This implies that the polygon is 1-fillable by Theorem 4.2.1. Thus, one venting hole is necessary.

Inductive Hypothesis: Assume that the number of venting holes needed = the number of local maximum vertices and edges, when $max \leq k$, $k \geq 1$.

Inductive Step: Let $max = k + 1$. Suppose that polygon P has at least two local maximum vertices. The argument is similar for local maximum edges.

Either the clockwise chain or counter-clockwise chain between the two local maxima contains a reflex vertex v or reflex edge e such that the vertices adjacent to v lie above $h(v)$ or the vertices adjacent to e lie above $h(e)$. Without loss of generality, let us assume that there is a reflex vertex v.

Extend a horizontal ray from v to the right until it intersects the polygon boundary. Let i be the intersection point. The chord $[vi]$ partitions the polygon into two subpolygons, P_1 and P_2. Each has less than $k + 1$ local maximum vertices or edges by construction. Suppose P_1 has $w \geq 1$ local maximum vertices, then P_2 has $k + 1 - w$ local maximum vertices. Thus by the induction hypothesis, P_1 needs w venting holes and P_2 needs $k + 1 - w$ venting holes. This totals to $k + 1$ venting holes in P, as required.

($\Leftarrow$) Suppose one of the local maxima in P did not have a venting hole or pin gate. Let v_i represent that local maximum in P. Let x be the farthest point from v_i on $[v_i v_{i+1}]$ visible from v_{i-1}. Triangle(v_{i-1}, v_i, x) contains neither a venting hole nor a pin gate. Therefore it is an air pocket. ■

Theorem 4.2.2 *Given a simple polygon in a fixed orientation, in $O(n)$ time one can determine the minimum k for which the polygon is k-fillable.*

Let us turn our attention to polygons with holes. A polygon with holes is defined as a polygon P enclosing several other polygons $H_1, \ldots, H_k$, the holes, such that none of the boundaries of $P, H_1, \ldots, H_k$ intersect and each of the holes is a simply connected region. A polygon with holes is an object that can be constructed with cores and inserts [47], [28]. In a polygon with holes, the peak of an air pocket may not involve a vertex or edge of the polygon, but a vertex or edge of a hole (see Figure 4.4). Therefore, there are two types of peaks of air pockets that may exist in a polygon with holes. The first type was described in Observation 4.2.1. The other type is described in the following observation.

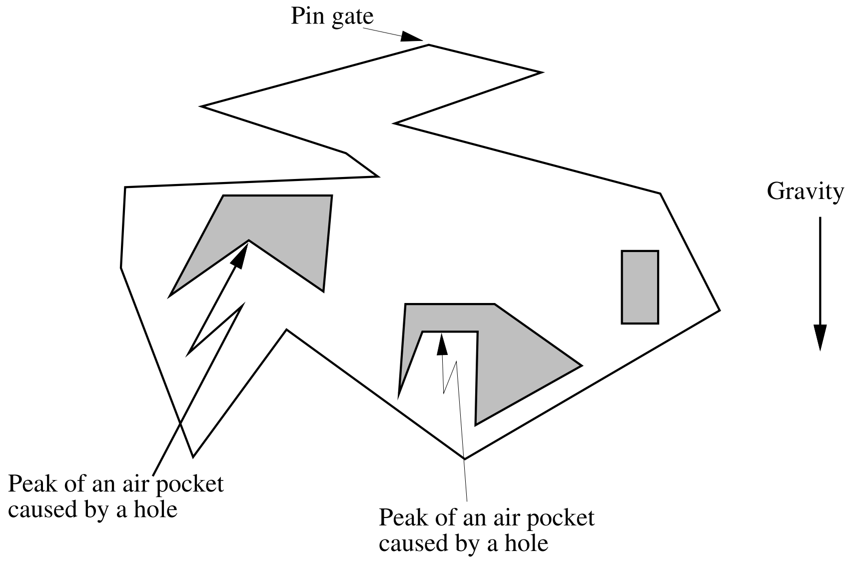

Figure 4.4: 1-fillable polygon that is no longer 1-fillable because of holes.

Observation 4.2.2 *A reflex vertex v or a reflex edge e of a hole H_j in polygon P is a peak of an air pocket provided that it is a local maximum of $bd(H_j)$ with respect to the direction of gravity (see Figure 4.4). Such a vertex or edge will be referred to as a* reflex maximum vertex *or* reflex maximum edge*.*

This observation provides a characterization of the peaks of air pockets caused by the presence of holes in a simple polygon. The characterization is similar to that of the peaks of air pockets in simple polygons without holes. Therefore, we have the following theorem.

Theorem 4.2.3 *A polygon with holes is 1-fillable if and only if the polygon has only one local maximum vertex or edge with respect to the direction of gravity and none of the holes have a reflex maximum vertex or reflex maximum edge.*

Proof: Similar to the proof of Theorem 4.2.1. ■

Corollary 4.2.2 *A polygon P with holes is 1-fillable if and only if $\forall p \in P$, the shortest path from p to the pin gate is monotonic with respect to the direction of gravity.*

Similar to the case of simple polygons without holes, to determine if a simple polygon with holes is k-fillable, simply test with respect to the direction of gravity, every convex vertex and convex edge of the polygon to determine whether or not it is a local maximum and test every reflex vertex and reflex edge of the holes to determine whether or not it is a reflex maximum. Since testing a vertex or edge can be done in constant time, we have the following theorem.

Theorem 4.2.4 *Given a simple polygon with holes and the direction of gravity, one can determine the minimum k for which the polygon is k-fillable in $O(n)$ time where n is the number of vertices of the polygon and the holes.*

4.3 Determining all Directions of Fillability

In the previous section, we showed that given a simple polygon without holes and the direction of gravity, we can determine in linear time the minimum k for which the polygon is k-fillable with respect to gravity. The extension to polygons with holes was immediate from Observation 4.2.2, so for simplicity of exposition, we continue the discussion with simple polygons without holes.

Suppose that we are given a polygon and asked whether there exists an orientation of the polygon such that the polygon is 1-fillable with respect to the direction of gravity. For example, the polygon in Figure 4.5 is 1-fillable in one orientation but not in another. In this section, we show that in $O(n \log n)$ time the complete range of directions of gravity that allow the 1-fillability of a polygon can be determined. In fact, in optimal $\Theta(n \log n)$ time we can determine all the orientations of the polygon that allow it to be k-filled where k is minimum over all orientations.

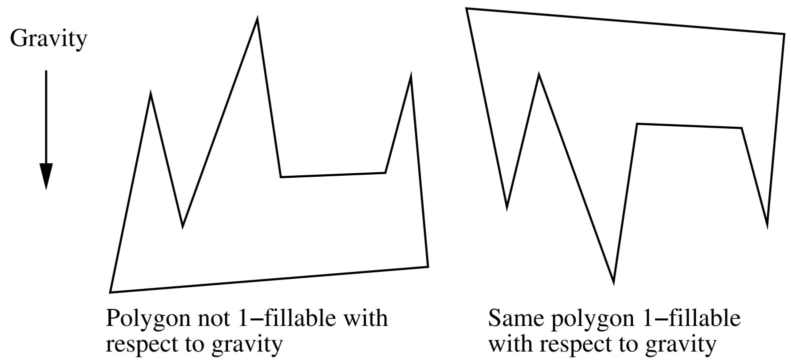

Figure 4.5: A polygon that is 1-fillable from one orientation but not another.

Let us examine the set of directions that cause a convex vertex to be a local maximum. Given a convex vertex v_i of polygon P, the set of directions for which v_i is a local maximum is defined as follows and will be denoted by $M(v_i)$ (refer to Figure 4.6).

$$M(v_i) = \{\forall \text{ directions } d \parallel ray(v_i v_{i+1}) \in NH(d) \text{ and } ray(v_i v_{i-1}) \in NH(d)\}$$

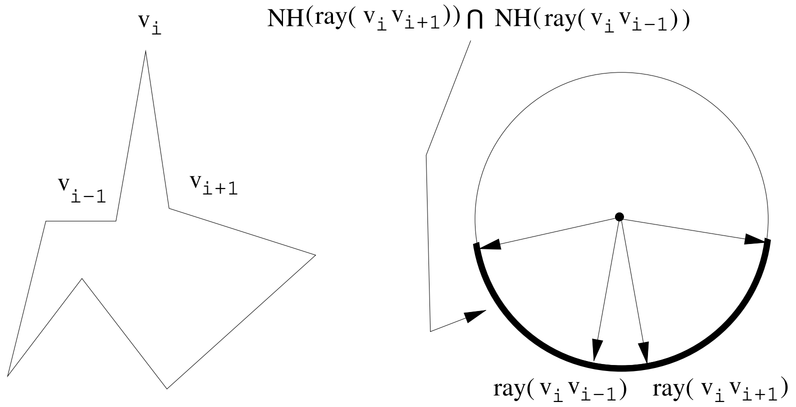

Figure 4.6: When a convex vertex is a local maximum.

In the following lemma, we characterize the directions in the set $M(v_i)$.

Lemma 4.3.1 $M(v_i) = NH(ray(v_i v_{i-1})) \cap NH(ray(v_i v_{i+1}))$

Proof: Let $r_{i-1} = ray(v_i v_{i-1})$ and $r_{i+1} = ray(v_i v_{i+1})$. For v_i to be a local maximum with respect to a given direction d, both r_{i-1} and r_{i+1} must be in $NH(d)$. Every direction $\theta \in NH(r_{i-1}) \cap NH(r_{i+1})$ is contained in $M(v_i)$ since both r_{i-1} and r_{i+1} are contained in $NH(\theta)$. Therefore let us consider an arbitrary direction $y \in M(v_i)$ that is not in $NH(r_{i-1}) \cap NH(r_{i+1})$. Then, one of r_{i-1} or r_{i+1} is not contained in $NH(y)$, contradicting the fact that v_i is a local maximum with respect to y. ■

A convex edge can be a local maximum edge only when the direction of gravity is orthogonal to the line containing the edge. Therefore, the proofs of Lemma 4.2.1 and Lemma 4.3.1 suggest the following algorithm to find the minimum number of venting holes needed to fill a polygon given that it must be filled in only one orientation.

Algorithm 5: *Find orientation minimizing number of venting holes.*

1. Find the direction that minimizes the number of local maximum vertices and edges.
2. Place the pin gate at the global maximum. [This can be done in $O(n)$ time].
3. Place a venting hole at every local maximum that is not a global maximum. [This can be done in $O(n)$ time by just scanning the boundary of the polygon].

Let us elaborate on the first step. Each convex vertex has an open arc representing the set of directions that cause that vertex to be a local maximum. If vertex v_i is convex, we denote the arc by $arc(a_i, b_i)$. Let A be the set of endpoints of all the arcs. Similarly, each convex edge has a point representing the direction that causes it to be a local maximum. If edge e_i is convex, let p_i represent this point. Let E be the set of all of these points. Pick an arbitrary direction $d \notin E$, and radially sort $A \cup E$ in a clockwise manner with respect to d. Let $S = s_1, s_2 \ldots s_m$ represent this sorted order.

We now perform a rotational sweep to determine the set of directions that minimizes the number of local maxima. Let c be the number of vertices that are local maxima with respect to d. Consider the first element of the sequence S. If s_1 is the end of an arc, then we know that the directions represented by $arc[d, s_1)$ have c local maxima and at direction s_1 have $c - 1$ maxima. If s_1 is the beginning of an arc, then we know that the directions represented by $arc[d, s_1]$ have c local maxima and the directions after s_1 have $c + 1$ local maxima. Finally, if s_1 is a point of E, then we know that the directions represented by $arc[d, s_1)$ have c local maxima, and at direction s_1 there are $c+1$ local maxima. By proceeding in this manner, the intervals on the unit circle induced by the set S are labeled with the number of local maxima present for each interval of directions. By choosing all the intervals with the smallest number, we have a complete description of all the directions from which the given polygon can be filled with a minimum number of venting holes. Some care must be taken when the endpoints of arcs and points coincide. The details of this technique may be found in [49]. The time complexity of this step is $O(n \log n)$ due to sorting.

The correctness of the algorithm follows from Lemma 4.2.1 and Lemma 4.3.1. The time complexity of the algorithm is dominated by step 1. Therefore, the total time complexity of the algorithm is $O(n \log n)$.

Theorem 4.3.1 *The minimum number of venting holes needed to fill a simple polygon from one fixed direction can be computed in $O(n \log n)$ time.*

In [35], it was shown that an $\Omega(n \log n)$ lower bound exists for the problem of determining the minimum number of venting holes to fill a simple polygon from one fixed direction by a reduction from *Element Uniqueness*.

Notice that the technique used in Algorithm 1 is not restricted to finding an orientation of a polygon that minimizes the number of local maxima. A local minimum can only be a convex vertex or convex edge, and the directions that cause such a vertex or edge are defined similarly to the directions that cause them to be local maxima. Therefore, this technique can be used to maximize the number of local maxima, minimize the number of local minima, maximize the number of local minima, minimize the combined number of local minima and maxima and maximize the combined number of local minima and maxima.

Theorem 4.3.2 *Given a simple polygon, in $O(n \log n)$ time, we can find the set of directions that minimize or maximize the number of local minima, the number of local maxima or the combined number of local minima and maxima.*

4.4 Fillability of Certain Classes of Polygons

As there is an $\Omega(n \log n)$ lower bound for determining the orientation minimizing the number of venting holes needed to fill a simple polygon, we study the relationship between certain known classes of polygons and fillability. We show that for some restricted classes of polygons, the optimal orientation for filling can be determined in linear time.

4.4.1 Monotone Polygons

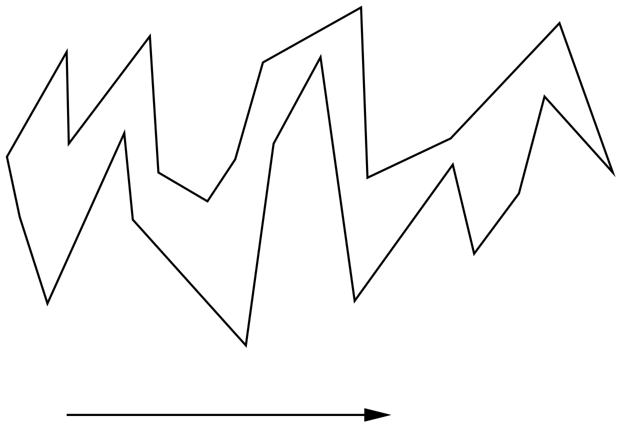

Figure 4.7: Monotone polygon.

A simple polygon P is a *monotone polygon* if there exists a line $L(\Theta)$ such that the boundary of P can be partitioned into two chains $C_{ij}(P)$ and $C_{ji}(P)$ that are monotonic with respect to Θ. (Refer to Figure 4.7).

Theorem 4.4.1 *A monotone polygon is 1-fillable if it is oriented such that gravity is a direction of monotonicity.*

Proof: Let P be a monotone polygon oriented such that g represents a direction of monotonicity and the direction of gravity. Without loss of generality, let this direction be the negative y direction.

If P is not 1-fillable, then by Theorem 4.2.1, it must contain at least two local maxima with respect to the direction of gravity. Between these two local maxima, there exists a reflex vertex v or reflex edge e such that the vertices adjacent to v lie above $h(v)$ or the vertices adjacent to e lie above $h(e)$. The existence of such a vertex or edge violates monotonicity. Therefore, a monotone polygon is 1-fillable. ∎

Since monotone polygons can be recognized in linear time [70] and the direction of monotonicity delivered as a witness, Theorem 4.4.1 provides a linear time algorithm for determining the optimal orientation of simple polygons if they are monotone.

4.4.2 Weakly-Edge Visible Polygons and Star-Shaped Polygons

Two points inside a polygon are said to be *visible* if the line segment between them does not intersect the exterior of the polygon. A point p is *weakly visible* from an edge e if there is a point x on e such that p is visible from x.

A polygon P is *edge visible* if there is an edge in the polygon from which all the points in the polygon are weakly visible. A polygon P is *open-edge visible* if there is an edge e in P such that all points p are visible from some point x on e other than the endpoints of the edge.

Let P be an open-edge visible polygon. Without loss of generality, let $(v_1 v_2)$ be the open edge from which the polygon is weakly visible. Let the polygon be oriented such that gravity g^* is the clockwise normal to *ray*$(v_1 v_2)$. (Refer to Figure 4.8).

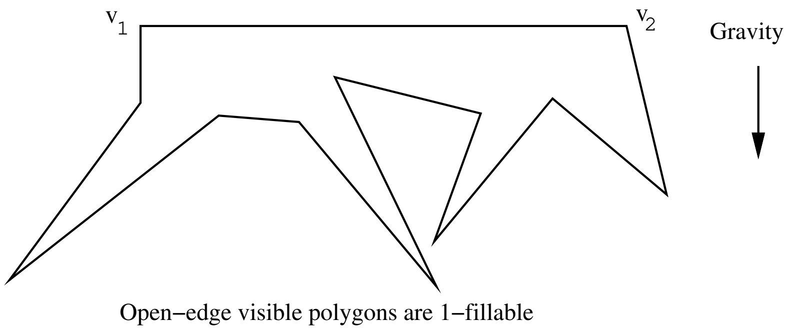

Figure 4.8: Open-edge visible polygon.

Theorem 4.4.2 *An open-edge visible polygon P with gravity g^* is 1-fillable.*

Proof: For ease of exposition, let us assume that gravity is in the negative y-direction. Without loss of generality, let v_1 be the pin gate. Let p be an arbitrary point in P. Since P is open-edge visible, there must be a point z on (v_1v_2) that sees point p, i.e. $[pz] \in P$.

Let Π be the shortest path from p to v_1 in polygon P. Since $[v_1p]$ is monotone with respect to gravity, and Π is a convex chain from p to v_1 contained in the triangle(z, p, v_1), Π is monotone with respect to gravity. Therefore, the lemma follows from Corollary 4.2.1. ■

Since open-edge visible polygons can be recognized in linear time [7], [74] and the required edge delivered as a witness, Theorem 4.4.2 provides a linear time algorithm for determining the optimal orientation of simple polygons if they are edge-visible.

Corollary 4.4.1 *Any polygon that is weakly visible from a chord is 2-fillable with re-orientation.*

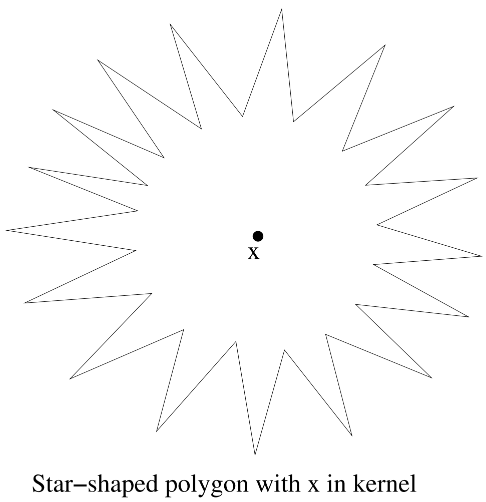

Figure 4.9: Star-shaped polygon.

A *star-shaped polygon* is a polygon that contains at least one point x from which all points of the polygon are visible. The set of points from which all points are visible is known as the *kernel* of the star-shaped polygon. The kernel of a star-shaped polygon can be computed in $O(n)$ time using the algorithm of Lee and Preparata [50] or a point in the kernel of a star-shaped polygon can be computed in $O(n)$ time using Megiddo's linear programming technique [57]. This implies that in $O(n)$ time, a chord can be found from which the star-shaped polygon is weakly visible. However, a star-shaped polygon may not necessarily be 1-fillable. The star-shaped polygon in Figure 4.9 is not 1-fillable since there are always two local maxima with respect to every direction. Therefore, we have the following.

Theorem 4.4.3 *A star-shaped polygon is not necessarily 1-fillable but is always 2-fillable with re-orientation in $O(n)$ time.*

4.4.3 Clam-Shell Polygons

A polygon is *clam-shell* if it can be partitioned into two chains $C_{ij}(P)$ and $C_{ji}(P)$ such that each chain can be removed from the mold by a single translation (not necessarily in a common direction). In Figure 4.10, we have a polygon that is not clam-shell. Clam-shell polygons were studied in [73] where the following result was proved. (Refer to Figure 4.11).

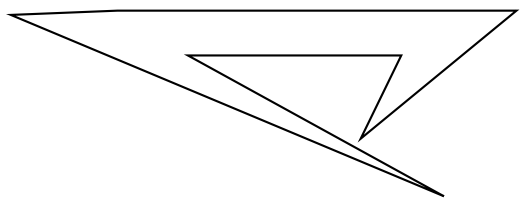

Figure 4.10: Simple polygon that is not clam-shell.

Theorem 4.4.4 *[73] A polygon is clam-shell if and only if the boundary can be decomposed into two chains, each monotonic to an arbitrary direction. Clam-shells can be recognized in linear time.*

Thus we see that this class is a generalization of monotonic polygons. Before showing that all clam-shell polygons are 1-fillable, we establish some key properties of clam-shell polygons. Let $C_{ij}(P)$ be monotonic with respect to some direction Θ. The monotonicity of $C_{ij}(P)$ implies that the polygon P has only one local maximum on the chain $C_{ij}(P)$ for all directions in $arc[\Theta, opp(\Theta)]$. Therefore, we have the following theorem.

Theorem 4.4.5 *A clam-shell polygon is 1-fillable.*

Proof: Given a clam-shell polygon P, let $C_{ij}(P)$ be monotone with respect to direction Θ_1 and $C_{ji}(P)$ be monotone with respect to direction Θ_2. The intersection $arc[\Theta_1, opp(\Theta_1)] \cap arc[\Theta_2, opp(\Theta_2)]$ must be non-empty since both arcs are closed semi-circles. ∎

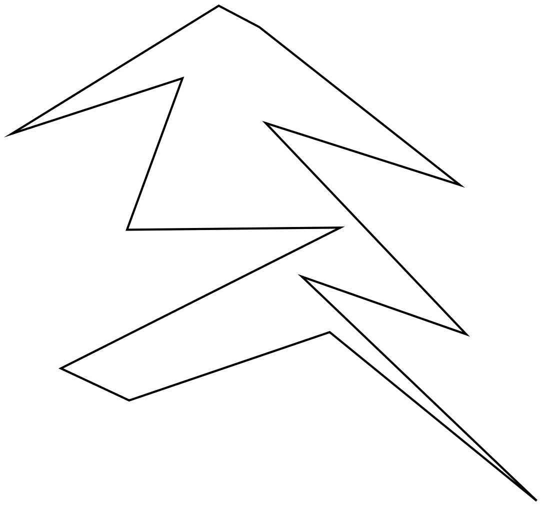

Figure 4.11: Clam-shell polygon.

Since clam-shell polygons can be recognized in linear time [73] and the required partition of the boundary delivered as a witness, Theorem 4.4.5 provides a linear time algorithm for determining the optimal orientation of simple polygons if they are clam-shell.

4.4.4 L-Convex Polygons

A polygon P is *L-convex* if $\forall x, y \in$P$, \exists z \in P$ such that $[xz] \in P$ and $[yz] \in P$. (Refer to Figure 4.12). From the definition of L-convexity, we see that a star-shaped polygon is L-convex since all points are seen by a kernel point.

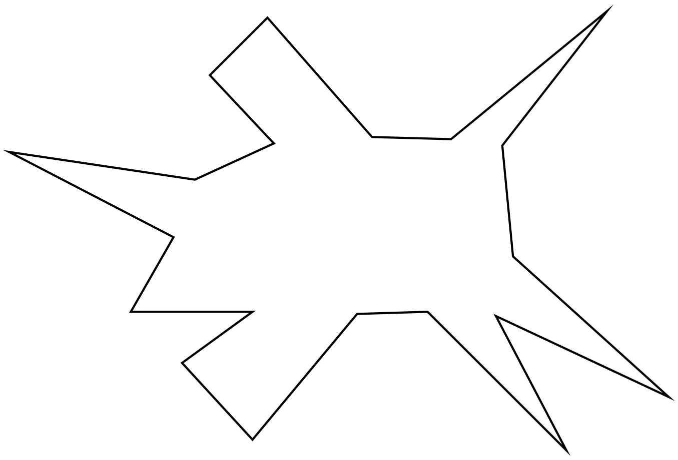

Figure 4.12: L-convex polygon.

Lemma 4.4.1 *[43] If P is an L-convex polygon, it has the property that for every point $x \in P$, there exists a chord of the polygon containing x from which P is weakly visible.*

The lemma proved in [43] is more general and applies to L-convex sets. From Corollary 4.4.1 together with the example of the star-shaped polygon that is not 1-fillable (Figure 4.9), we have the following.

Theorem 4.4.6 *An L-convex polygon is not necessarily 1-fillable but always 2-fillable with re-orientation.*

Lemma 4.4.1 only proves the existence of the chord, but does not offer a method of computing such a chord given an L-convex polygon. We now present an $O(n \log n)$ time algorithm to find such a chord. We first prove a few key lemmas.

Let x be a point inside polygon P. The *visibility polygon* from x, denoted by $VP(x, P)$, is the set of points in P visible from x. It is formally defined as $VP(x, P) = \{z \mid z \in P \text{ and } [xz] \cap P = [xz]\}$ Let v_i be an arbitrary vertex of an L-convex polygon P. By Lemma 4.4.1, there must be a chord containing v_i from which P is weakly visible. Let us denote this chord by $C(v_i)$. Thus, $C(v_i)$ must be contained in $VP(v_i, P)$. We will now show the relationship between a diagonal in $VP(v_i, P)$ and $C(v_i)$. Let b be a vertex of $VP(v_i, P)$. The chord $[v_i b]$ is a chord in P and divides the polygon into two subpolygons, P_1 and P_2. If both P_1 and P_2 are weakly visible from $[v_i b]$, then we have found $C(v_i)$. Otherwise, we will show that either P_1 or P_2 has to be weakly visible from $[v_i b]$.

Lemma 4.4.2 *Either P_1 or P_2 or both are weakly visible from $[v_i b]$.*

Proof: If both are weakly visible from $[v_i b]$ then the lemma holds. Suppose that one of P_1 or P_2 is not weakly visible from $[v_i b]$.

Case 1: $C(v_i) \in P_2$. By Jordan's Curve Theorem, every line segment with one endpoint in P_1 and one endpoint in P_2 must intersect $[v_i b]$. P_1 is weakly visible from $C(v_i)$. This means $\forall x \in P_1, \exists y \in C(v_i)$ such that $[xy] \in P$. But then $[xy]$ must intersect $[v_i b]$. Thus, P_1 is weakly visible from $[v_i b]$.

Case 2: $C(v_i) \in P_1$. Symmetric to case 1. ∎

Notice that if P_1 is not weakly visible from $[v_i b]$ then $C(v_i)$ is contained in P_1 and vice versa. Thus, $C(v_i)$ is either a diagonal of the polygon P or contained between two consecutive diagonals. The question now becomes, how do we compute $C(v_i)$ if it is contained between two consecutive diagonals?

Consider the following situation. Let r and l be two consecutive vertices in $VP(v_i, P)$ such that P_1 is weakly visible from $[v_i l]$ but not $[v_i r]$ and P_2 is weakly visible from $[v_i r]$ but not $[v_i l]$.

Let $a_1, a_2, \ldots, a_m$ be the vertices in P_1 that are not weakly visible from $[v_i r]$. They must be visible from $[lr]$ since $C(v_i)$ is contained in the triangle(v_i, l, r). Similarly, let $b_1, b_2, \ldots, b_t$ be the vertices of P_2 that are not weakly visible from $[v_i l]$. They must also be visible from $[lr]$. Let a'_i be the point on $[lr]$ farthest away from l, from which a_i is visible. Let $\overline{a} = \min_{1 \leq i \leq m} \; |[la'_i]|$. Similarly, let b'_i be the point on $[lr]$ farthest away from r, from which b_i is visible. Let $\overline{b} = \min_{1 \leq i \leq t} \; |[lb'_i]|$. Let $S = \overline{a} \cap \overline{b}$.

Lemma 4.4.3 *S is not empty.*

Proof: Suppose S was empty. We know that $C(v_i) \in$ triangle(l, r, v_i). Now, if $C(v_i) \cap [lr] \notin \overline{b}$, then there would be some b_i that was not weakly visible from $C(v_i)$ which is a contradiction. Similarly, if $C(v_i) \cap [lr] \notin \overline{a}$, then there would be some a_i that was not weakly visible from $C(v_i)$ which is a contradiction. ∎

Lemma 4.4.4 *For every $s \in S$, the polygon P is weakly visible from $[v_i s]$.*

Proof: If a point $p \in P$ is weakly visible from both $[v_i l]$ and $[v_i r]$ then it must also be weakly visible from $[v_i s]$ since any line segment that intersects both $[v_i l]$ and $[v_i r]$ also intersects $[v_i s]$.

If p is not weakly visible from $[v_i r]$ then let p' be the farthest point from l from which p is weakly visible. $\overline{a}$ must be contained in $[lp']$. Thus, p must be weakly visible from $[v_i s]$. A similar argument shows that if p is not weakly visible from $[v_i l]$, it is still weakly visible from $[v_i s]$. ■

Lemma 4.4.2 and Lemma 4.4.4 suggest the following algorithm.

Algorithm 6: *Compute weakly visible segment*

1. Choose a vertex $v_1 \in P$.
2. Compute $VP(v_1, P)$ in $O(n)$ time using the algorithm of Avis and Elgindy [30]. [Let $\{v_1, \ldots, v_k\}$ denote the vertices of $VP(v, P)$]
3. Triangulate $VP(v, P)$ by inserting all diagonals $d_i = [v_1 v_i]$, $2 \leq i \leq k$.
4. Let $l = 2$, $r = k$.
5. Let $s = \lfloor r/2 \rfloor$
6. Let P_1 be the polygon with vertices $\{v_1, v_l, v_{l+1}, \ldots, v_s\}$
7. Let P_2 be the polygon with vertices $\{v_1, v_s, v_{s+1}, \ldots, v_r\}$
8. If both P_1 and P_2 are weakly visible from d_s, exit with d_s. [Use the algorithm of Avis and Toussaint [7] to verify weak visibility from d_s in $O(n)$ time].
9. If $r - l = 1$ go to step 12. [two consecutive diagonals at this point].
10. If P_1 is not visible from d_s, then $r = s$, go to step 5.
11. If P_1 is not visible from d_s, then $l = s$, go to step 5.
12. Compute $\overline{a} \cap \overline{b}$ in $O(n)$ time using the algorithm of Avis and Toussaint [7]. Pick any point s in $\overline{a} \cap \overline{b}$ and exit with $[v_1 s]$.

The correctness of the algorithm follows from the discussion. Each step takes at most $O(n)$ time and we loop through steps 5 to 11 at most $O(\log n)$ times. Thus the total time complexity of the algorithm is $O(n \log n)$.

4.4.5 Weakly-Externally Visible Polygons

We have seen that the class of clam-shell polygons are 1-fillable and the class of star-shaped polygons are 2-fillable. We have also seen that the class of L-convex polygons, which contains the class of star-shaped polygons is also 2-fillable. A natural question is whether the class of weakly-externally visible polygons is 2-fillable, since the class of weakly-externally visible polygons contains the class of L-convex polygons (this is shown in [43]) and the class of clam-shell polygons. A polygon is weakly-externally visible if for every point x on its boundary there is an infinite ray emanating from that point in some direction that intersects the boundary only at x. In fact, the class of weakly-externally visible polygons is in some sense the largest class of interest since it is the largest class of polygons that allows the pin gate to be placed anywhere on its boundary, if we require the pin gate to be reachable by a line probe from infinity. Consider the weakly-externally visible polygon in Figure 4.13. No matter what direction we pour it from, at most only 1/4 of the arms will be completely filled. But, 3/4 of the arms cannot be filled from a second pin gate and orientation. Therefore, we have the following theorem.

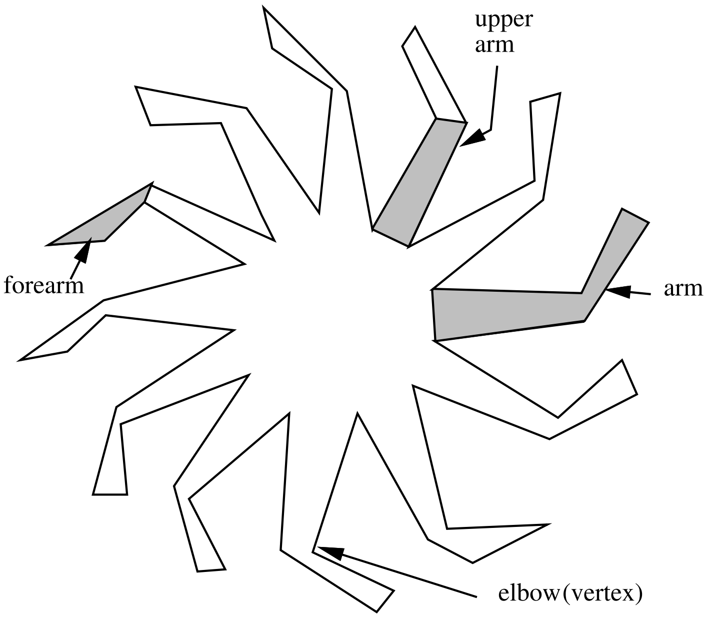

Figure 4.13: Weakly-externally visible polygon.

Theorem 4.4.7 *A weakly-externally visible polygon is not necessarily 2-fillable with re-orientation.*

We summarize the relation between fillability and some known classes of polygons by the chart in Figure 4.14.

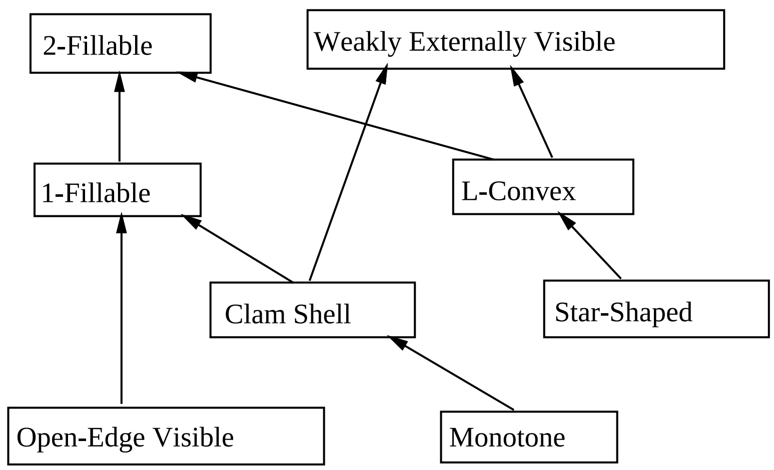

Figure 4.14: Summary of relation between fillability and classes of polygons.

Chapter 5

Gravity Casting in Three Dimensions

5.1 Introduction

In this chapter, we generalize the tools and techniques of the previous chapter to handle molds modeled as simple polyhedra (see Figure 5.1). The conceptual approach is identical to the two dimensional case, however, the technical details involved in the three dimensional case are more complex, as expected.

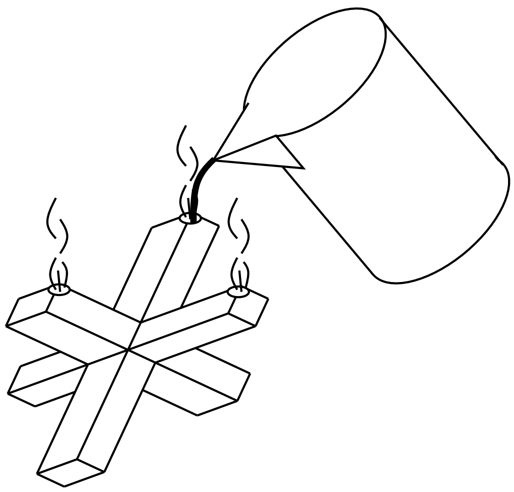

Figure 5.1: Gravity casting of a star-shaped object using one filling hole and two additional venting holes.

We show that given a mold, represented by a simple polyhedron with n vertices in a fixed orientation, we can determine in $O(n)$ time whether or not the mold can be filled without forming air pockets. Thus, the time complexity of the decision problem is the same whether we have polygonal molds or polyhedral molds. On the other hand, the time complexity of finding all orientations that allow a k-filling for minimum k for polygonal molds was shown to be $O(n \log n)$ in the previous chapter, but in this chapter, we are only able to achieve $O(n^2)$ for polyhedral molds. However, we are able to justify this increase in time complexity by providing a *pseudo-lower bound* for this problem. We reduce the problem '$A+B=C$?' to the three dimensional mold filling problem. The problem '$A+B=C$?' is defined as follows: Given three sets A, B, and C of n real numbers each, decide if there exists $a \in A$, $b \in B$ and $c \in C$ such that $a+b=c$. The best known algorithm for solving this problem uses $O(n^2)$ time. Gajentaan and Overmars [37] have shown there exist many problems in geometry that also reduce to '$A+B=C$?', such as: 'Given a set of n points in the plane, are there three collinear points?' and 'Given a set of n rectangles in the plane, do they cover a given rectangle $RECT$ completely?'. Since the best known algorithms take $O(n^2)$ time to solve any one of these problems, a problem which can be reduced to one of these is referred to as an n^2*-difficult* problem. Since the mold filling problem is n^2-difficult, improving on the quadratic bound seems difficult.

The interesting question that arises is whether one can improve the $O(n^2)$ time bound for some restricted classes of polyhedra. We relate fillability to certain known classes of polyhedra, namely, star-shaped, monotone, and facet-visible polyhedra. In the case of star-shaped polyhedra, this reduces the time bound for finding an optimal orientation to $O(n)$ time as opposed to $O(n^2)$ time.

5.2 Preliminaries

In this chapter, it will be convenient to have the set of all directions in space be represented by two planes. Although this is not standard, it will help simplify the exposition. Let the plane $z=-1$, denoted by $DP^{(-)}$, represent all directions with a negative z-component, and the plane $z=1$, denoted by $DP^{(+)}$, represent all directions with a positive z-component. We do not consider the horizontal directions. This assumption simplifies our discussion but is not an inherent limitation of our methods. A point q in $DP^{(-)}$ or $DP^{(+)}$ represents the direction $\vec{oq}$, where o represents the origin of E^3. Given a direction d, represented by $\vec{oq}$, we define $opp(d)$ to be the opposite direction. Thus, $opp(d)$ is pointing in the direction of the vector $\vec{qo}$.

A polygonal chain $C = p_0, p_1, \ldots, p_n$ is *monotonic* with respect to direction Θ if the projections of the vertices $p_0, p_1, \ldots, p_n$ onto a line in direction Θ are ordered as the vertices in C.

5.2.1 Geometric Model of Gravity Casting

We now generalize the gravity model. A mold is modeled by a simple polyhedron. The point on the boundary of a mold through which the liquid is poured into the polyhedron is called the *pin gate*. We assume that the pin gate is the only point from which air is allowed to escape unless stated otherwise. A *venting hole* is a point from which only air and no liquid is allowed to escape. We assume that neither the liquid being poured into the mold, nor the air in the mold are compressible. Finally, we assume that air cannot bubble out through the liquid.

The sole force acting on the liquid is gravity. When a direction of gravity is not specified, we assume, for simplicity of exposition, that gravity points in the negative z-direction. Thus, if only one pin gate is used, we assume it to be a point on the boundary with the highest z-coordinate, since otherwise, the polyhedron cannot be completely filled.

When liquid is poured into a polyhedron, the level of the liquid rises in the direction opposite that of gravity. We assume that the advancing *front* of the rising liquid is a plane. The lowest horizontal plane such that all the liquid in the polyhedron is contained below it, is defined as the *level plane*.

When the level plane contains the pin gate, we say the polyhedron is *maximally filled*. A region containing air in a maximally filled polyhedron is called an *air pocket*. A polyhedron is said to be *1-fillable* if there exists a pin gate and direction of gravity such that when the liquid is poured into the polyhedron through the pin gate, there are no air pockets when the polyhedron is maximally filled. We call the highest point (there may be more than one) of an air pocket in a maximally filled mold, the *peak* of the air pocket. This leads to the following observation.

Observation 5.2.1 *A polyhedron P in 3-space is said to be* 1-fillable in direction $-z$ *provided that for every point inside P there is a $+z$-monotone path from it to the z-maximum of P. Thus, if a polyhedron is* 1-fillable *there exists an orientation of P in which it is 1-fillable.*

We extend the notion of fillability in the following two ways. A polyhedron is said to be *k-fillable* if there exists a fixed orientation of the polyhedron, a placement of the pin gate and a placement of $k-1$ venting holes such that when liquid is poured into the polyhedron through the pin gate, there are no air pockets when the polyhedron is maximally filled. A polyhedron is said to be *k-fillable with re-orientation* provided that the polyhedron can be re-oriented and filled from a new pin-gate after partial filling from an initial orientation and pin gate. We assume that after the completion of a partial filling, the liquid that is poured into the polyhedron hardens. The number k in this case refers to the number of times that the polyhedron needs to be re-oriented before it is completely filled. Notice that both definitions are identical when $k = 1$. Unless stated otherwise, we will always refer to *k-fillable* as filling from a fixed orientation.

5.3 The Decision Problem

In this section we present an $O(n)$ time algorithm to decide whether a polyhedron P is 1-fillable given a fixed orientation of the polyhedron.

Let P be a simple polyhedron of which all facets are triangulated, and let v be an arbitrary vertex of P. We define P_v to be the union of the facets incident to v. Let $f_1, \ldots, f_m$ be the sequence of facets of P_v such that f_i and f_{i+1} are incident to an edge denoted e_i, and f_m and f_1 are incident to an edge e_m. Let S_v be a sphere centered at v, such that S_v only intersects the m edges incident to v, and no other facets, edges or vertices of P.

Definition 5.3.1 *A vertex v is a* convex vertex of P *provided that there exists a plane h_v, with $v \in h_v$, such that $S_v \cap h_v$ does not intersect the interior of P.*

Let h_v^- and h_v^+ denote the closed half-spaces below and above the plane h_v, containing the vertex v. Let $h_v^\diamond$ be the closed half-space bounded by the plane h_v with normal d, containing the vertex v and where $\diamond \in \{-, +\}$ is the opposite of the sign of the z-component in d. Recall that we assume, for simplicity, that d is not a horizontal direction.

Definition 5.3.2 *A vertex v is a* local maximum of P in direction d *provided that P_v lies in the closed half-space $h_v^\diamond$.*

We now prove the theorem used to establish the linear time decision algorithm.

Theorem 5.3.1 *A polyhedron P is 1-fillable if and only if the orientation of P has precisely one local maximum in direction $+z$.*

Proof: We assume that gravity is in the $-z$ direction. Suppose that P is 1-fillable, and suppose that P has more than one local z-maximum. Let q be a local z-maximum of P which is not the global z-maximum M of P. Let Π be any path from q to M. Since q is a local z-maximum, Π has negative value in its z-component when it leaves q, contradicting Observation 5.2.1

On the other hand, suppose that P has only one local z- maximum M, which must also be the global z-maximum of P. Let p be any point inside P, and let f be the facet of P hit by a ray emanating from p vertically upward. Let q be the vertex incident to this facet with maximum z-coordinate. Clearly, there is a $+z$-monotone path from p to q consisting of two segments. If $q = M$ we are done, otherwise q is not a local z-maximum, and it must be incident to an edge with endpoints q and q' such that q' has greater z-coordinate. We repeat the argument with q' for q until the path reaches M. ■

From this theorem, we see that given a polyhedron P and a direction of gravity g, to test 1-fillability of P with respect to g, we need only determine the number of local maxima with respect to gravity. We can determine if a vertex is a local maximum in time linear in the degree of the vertex [57]. This immediately gives us a linear time algorithm to determine whether or not a polyhedron is 1-fillable from a fixed orientation.

Theorem 5.3.2 *Given a polyhedron P, we can determine in $O(n)$ time whether or not the polyhedron is 1-fillable with respect to gravity.*

5.4 Determining all Directions of Fillability

In this section we will give an $O(n^2)$ time algorithm to find the orientation of a given polyhedron P that minimizes the number of venting holes needed in order to ensure a complete fill from a fixed orientation. This orientation is equivalent to the orientation that minimizes the number of local maxima. The algorithm has two stages. In the first stage, the fillability problem is transformed to a planar problem for a set of convex (possibly unbounded) polygons that cover the plane. In the second stage, the following problem is solved: Given a set of n convex polygons in the plane, find the point that is covered by a minimum number of them.

5.4.1 Transforming Fillability to Covering

Let P be a bounded polyhedron with n vertices, and assume that P is given by its incidence graph (see e.g. [28]). First, we triangulate every facet of P (see e.g. [19, 69]). We choose an initial orientation of P such that no edge of P is vertical. Let v be any vertex of P. We extract the description of P_v from the description of P in time proportional to its size. Let $f_1, \ldots, f_m$ be the sequence of disjoint facets incident to v, such that f_i and f_{i+1} are incident to an edge e_i of P_v (and f_m and f_1 are incident to an edge e_m). Let $w_1, \ldots, w_m$ be the sequence of endpoints corresponding to $e_1, \ldots, e_m$, see Figure 5.2.

Suppose that v is a convex vertex. We define the *cone C_v of v* to be the unbounded polyhedron consisting of v as its only vertex, m half-lines $E_1, \ldots, E_m$ starting at v, which contain the edges $e_1, \ldots, e_m$, respectively, and m unbounded facets bounded by E_i and E_{i+1} $(1 \le i \le m-1)$, or E_m and E_1. Since C_v need not be a convex polyhedron, but its only vertex is convex, we say that C_v is a *semi-convex cone*. Let CC_v be the convex hull of C_v, which is a *convex cone*. The half-lines that are the edges of CC_v are a subset of the edges of C_v; we denote them by $E_{i_1}, \ldots, E_{i_j}$, where $1 \le i_1 < \cdots < i_j \le m$. Finally, we define the *normal cone* NC_v of the convex cone CC_v as follows. Let $h_{i_1}, \ldots, h_{i_j}$ be the set of planes that pass through v and are perpendicular to $E_{i_1}, \ldots, E_{i_j}$. Let $H_{i_1}, \ldots, H_{i_j}$ be the closed half-spaces bounded by $h_{i_1}, \ldots, h_{i_j}$ such that they contain $E_{i_1}, \ldots, E_{i_j}$, respectively. Then NC_v is the convex region that is bounded by $H_{i_1} \cap \cdots \cap H_{i_j}$. Notice that if CC_v is a sharp cone then NC_v is a blunt cone, and vice versa.

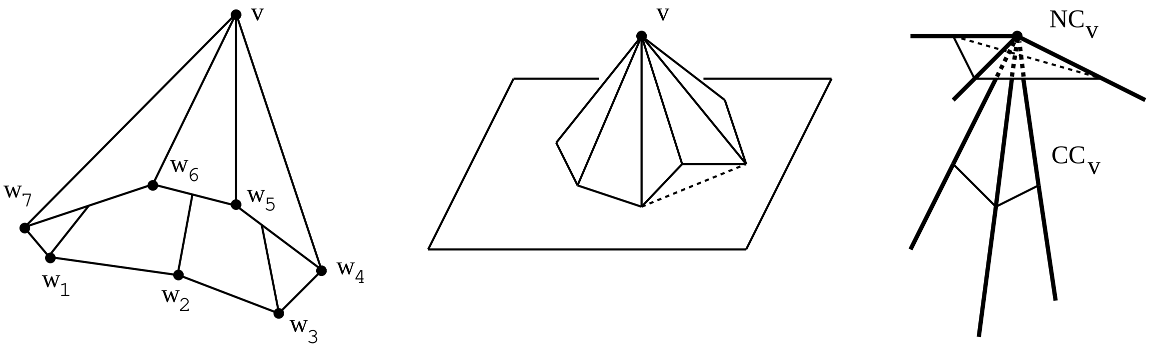

Figure 5.2: Left: P_v. Middle: the convex hull CC_v of C_v. Right: the convex cone CC_v and the normal cone NC_v.

Each convex vertex of the polyhedron P defines a convex region in $DP^{(-)}$ or $DP^{(+)}$ or both, which corresponds to the directions with respect to which it is a local maximum. Hence, P gives rise to $O(n)$ convex regions in these planes. It follows that a direction for which P has the smallest number of local maxima corresponds to some point in the plane that is covered by the smallest number of convex regions. The following lemma relates the normal convex cones to the direction planes, $DP^{(-)}$ and $DP^{(+)}$.

Lemma 5.4.1 *For every convex vertex v of a polyhedron P such that v coincides with the origin o and direction $d = \vec{oq}$ where q is a point on one of the direction planes, it holds that v is a local maximum in (non-horizontal) direction $-d$ if and only if $q \in NC_v \cap DP^{(-)}$ or $q \in NC_v \cap DP^{(+)}$.*

Proof: Let ℓ be the half-line rooted at o with direction d. By construction, the following equivalence holds for any convex vertex v located at o and $\diamond \in \{-, +\}$: There exists a plane h through v with normal d such that $CC_v \subseteq h^{\diamond}$ if and only if $\ell \subseteq \text{interior}(NC_v) \cup NC_v$. Since the direction d is represented by the point $q = \ell \cap DP^{(\diamond)}$, the lemma follows immediately. ∎

Therefore we first determine if v is a convex vertex. This is the case if and only if v is an extremal point in the set $\{v, w_1, \ldots, w_m\}$. This is equivalent to the problem of determining if v can be separated from $\{w_1, \ldots, w_m\}$ by a plane, which in turn is equivalent to linear programming [27]. Therefore we can determine if v is convex by linear programming in linear time (see e.g. [28, 57, 80]). If v is not a convex vertex, then v is not a local maximum for any direction, and we stop considering v. Otherwise, let h_v be a plane that contains v and has $w_1, \ldots, w_m$ to one side of it. Such a plane is returned by the linear programming test. Let h'_v be a plane parallel to h'_v which intersects all edges $e_1, \ldots, e_m$. The intersection of h'_v with P_v is a simple polygon $\bar{P}_v$ with m vertices (corresponding to $e_1, \ldots, e_m$) and m edges (corresponding to $f_1, \ldots, f_m$). We compute the convex hull of $\bar{P}_v$ in linear time [56], [59]. Let us denote the convex hull by $CH(\bar{P}_v)$. Let $\bar{e}_{i_1}, \ldots, \bar{e}_{i_j}$ be the sequence of vertices of $CH(\bar{P}_v)$, where $1 \leq i_1 < \cdots < i_j \leq m$. These vertices correspond to the edges $e_{i_1}, \ldots, e_{i_j}$ of P_v. We have in fact computed the edges adjacent to v on the convex hull of P_v. This information gives us the description of the convex cone CC_v of v in linear time. Furthermore, the normal cone NC_v can also be computed in additional linear time.

Translate NC_v such that v coincides with the origin o. Let $Q_v^{(-)}$ be the convex polygon $NC_v \cap DP^{(-)}$ and let $Q_v^{(+)}$ be $NC_v \cap DP^{(+)}$. Either $Q_v^{(-)}$ is a bounded convex polygon and $Q_v^{(+)}$ is empty, or vice versa, or both $Q_v^{(-)}$ and $Q_v^{(+)}$ are unbounded convex polygons. The convex polygons have the following meaning: v is a local maximum in a non-horizontal direction $-d$ if and only if the half-line starting at the origin o in direction d intersects the interior of one of the polygons $Q_v^{(-)}$ or $Q_v^{(+)}$. We compute the convex polygons $Q^{(-)}$ and $Q^{(+)}$ for all vertices of P, giving sets $\mathcal{Q}^{(-)}$ and $\mathcal{Q}^{(+)}$ of at most n convex polygons in the planes $DP^{(-)}$ and $DP^{(+)}$, respectively. The total complexity of the polygons in $\mathcal{Q}^{(-)}$ and $\mathcal{Q}^{(+)}$ is $O(n)$. The question: 'Is P 1-fillable?' or 'Is there an orientation of P such that it has only 1 maximum?' translates to the question: 'Is there a point in $DP^{(-)}$ or $DP^{(+)}$ that is covered by only one convex polygon?' Similarly, the question of k-fillability translates to deciding whether there exists a point that is covered by only k convex polygons. We therefore have established the following result:

Lemma 5.4.2 *In $O(n)$ time, the problem of k-fillability can be transformed to the problem of finding a point in the plane covered by only k convex polygons.*

5.4.2 Solving the Covering Problem

The next step in the algorithm involves solving the following problem: 'Given a set $\mathcal{Q}$ of n convex, but not necessarily bounded, polygons in the plane, with total complexity $O(n)$, find a point that is covered by the minimum number of polygons of $\mathcal{Q}$.' Our algorithm constructs the subdivision induced by $\mathcal{Q}$, and associates to each cell the number of polygons that contain it.

The subdivision induced by $\mathcal{Q}$ without the numbering can be constructed deterministically in $O(n \log n + A)$ time by the algorithm of Chazelle and Edelsbrunner[20], where A is the total number of intersection points of all polygons in $\mathcal{Q}$. Alternatively, a simpler randomized algorithm performs the task with the same time bound, see Clarkson[21] or Mulmuley[61]. The size of A can be $O(n^2)$. Therefore, we obtain a planar subdivision $\mathcal{S}$ with $O(n^2)$ vertices, edges and cells. Consider the graph G which has a node for every cell of $\mathcal{S}$, and an edge between two nodes if the corresponding cells are incident to the same edge of $\mathcal{S}$. The graph G has $O(n^2)$ nodes and edges. Start at any node a_1, and compute in $O(n)$ time how many polygons of $\mathcal{Q}$ cover it. Store this number with a_1. Start from a_1 with a depth first search. Every edge (a_i, a_j) of G we traverse corresponds to going inside or outside a polygon of $\mathcal{Q}$, in which case we take the number of a_i, add or subtract one from it, and assign this number to a_j. Thus the whole process of assigning values to cells of $\mathcal{S}$ requires only $O(n^2)$ time. The cell with the minimum number assigned to it is covered by the minimum number of polygons.

Returning to the k-fillability problem, the above algorithm finds the direction d such that the polyhedron has the minimum number of local maxima, if we apply it to both the set $\mathcal{Q}^{(-)}$ of convex polygons in the plane $DP^{(-)}$ and $\mathcal{Q}^{(+)}$ in the plane $DP^{(+)}$. We summarize the algorithm below.

Algorithm 7: *Find all orientations such that P is fillable with minimum number of venting holes.*

1. Select all convex vertices of polyhedron P.
2. Compute the convex cone of each convex vertex.
3. Compute the normal cone of each convex cone. Call this set NC.
4. Intersect each normal cone in NC with $DP^{(+)}$ and $DP^{(-)}$. Call this set of (possibly unbounded) convex polygons R.
5. Compute the arrangement $\mathcal{Q}^{(+)}$ induced by R on $DP^{(+)}$ and $\mathcal{Q}^{(-)}$ induced by R on $DP^{(-)}$.

6. Find all regions on $\mathcal{Q}^{(+)}$ and $\mathcal{Q}^{(-)}$ covered by the least number of convex polygons of the set R. These regions represent the orientations minimizing the number of venting holes need to fill P.

We conclude with the following theorem.

Theorem 5.4.1 *Given a simple bounded polyhedron P in 3-space, one can find in $O(n^2)$ time an orientation for P such that P is fillable with the minimum number of venting holes.*

5.5 A Reduction from Covering to 1-Fillability

In this section, we present an $O(n \log n)$ time reduction from the rectangle covering problem to the problem of 1-fillability of polyhedra. Since a reduction from the '$A+B=C$?'problem to rectangle covering is given in [37], it follows that 1-fillability is at least at hard as '$A+B=C$?'.

Theorem 5.5.1 *The rectangle covering problem can be reduced to the 1-fillability problem in $O(n \log n)$ time.*

Proof: Let I be an instance of the rectangle covering problem, i.e., given a set $\mathcal{R}$ of n rectangles in the plane, and also a rectangle $RECT$, decide if the union of the rectangles in $\mathcal{R}$ cover $RECT$. We now describe the construction of a polyhedron P such that it is 1-fillable if and only if the rectangle $RECT$ is not covered by $\mathcal{R}$.

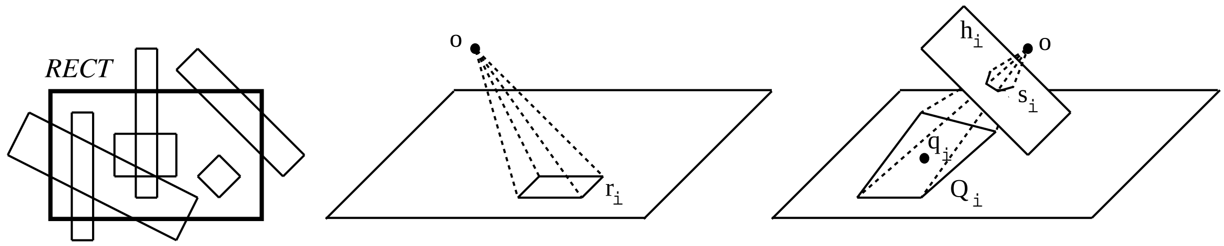

Figure 5.3: Left: an instance of the rectangle covering problem. Middle: a rectangle r_i and its convex cone $CC(r_i)$. Right: the normal convex cone $NC(r_i)$ and the spike s_i.

We associate the plane in which $\mathcal{R}$ and $RECT$ lie with the plane $z = -1$, such that the center of $RECT$ is the point $(0, 0, -1)$. For every $r_i \in \mathcal{R}$, we associate the convex cone $CC(r_i)$ to be the cone with apex the origin o of 3-space, and whose intersection with the plane $z = -1$ is the rectangle r_i. Then we normalize $CC(r_i)$ to obtain a convex cone $NC(r_i)$, and we intersect $NC(r_i)$ with the plane $z = -1$ to obtain a possibly unbounded convex polygon Q_i. For each Q_i, we choose a point q_i in its interior such that all of the q_i are distinct. (The convex hull of the q_i should contain the point $(0, 0, -1)$; if not, we add suitably chosen dummy rectangles to $\mathcal{R}$ outside of $RECT$ to enforce this.) Let h_i be the plane through o with normal $\vec{oq_i}$. Translate h_i in direction $\vec{oq_i}$ by an amount such that the interior of $h_i \cap NC(r_i)$ has positive area, but is contained in a disk with diameter 1. Define the *spike* s_i to be the polyhedron $h_i^+ \cap NC(r_i)$. Translate h_i and the spike s_i simultaneously back in direction $\vec{q_i o}$, such that h_i passes through o again.

Let γ be the minimum distance between any two of the distinct points q_i. Let Γ be the maximum distance of any q_i to the origin o. Let S be a sphere centered at o with radius at least $2\Gamma/\gamma + 1$.

Translate every pair h_i and s_i in direction $\vec{q_i o}$ such that h_i is tangent to S ($S \subseteq h_i^-$). By the choice of the radius of S and the area of $h_i \cap NC(r_i)$ (the 'base' of the spike), no two spikes s_i and s_j intersect. Compute the convex polytope $P = (z \geq -1) \cap \bigcap_{1 \leq i \leq n} h_i^-$. By construction (the addition of dummy rectangles), P is a bounded convex polyhedron. To P, we add each spike s_i on the facet of P that lies in h_i. To finish the construction, we add one more gadget to the facet contained in the plane $z = -1$. This is the new spike s_{RECT} for $RECT$, which is translated in the $-z$-direction over a distance so that its topmost point penetrates the lower facet of P.

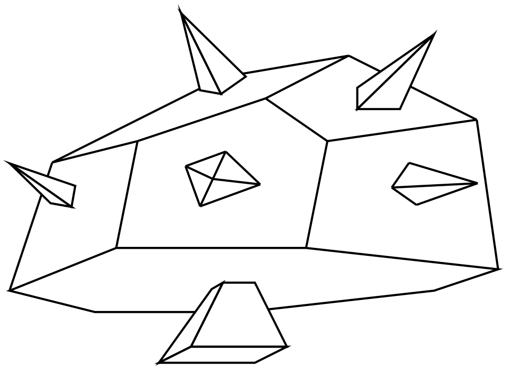

Figure 5.4: An example of the polyhedron constructed for theorem 5.5.1.

Without all the spikes, P is a convex polyhedron, and thus has exactly one maximum for every direction. The spike s_{RECT} gives additional local maxima for every direction corresponding to a point in $z = -1$ outside of $RECT$. The other spikes give a local maximum for every direction that corresponds to a point inside the corresponding rectangles of $\mathcal{R}$. Hence, P is 1-fillable if and only if $RECT$ is not covered by the union of the rectangles in $\mathcal{R}$. The construction can be performed in $O(n \log n)$ time using the half-space intersection algorithm of Preparata and Muller [68]. ∎

5.6 Fillability of Certain Classes of Polyhedra

In this section, we investigate the relationship between the notion of fillability and certain known classes of restricted polyhedra. These results are relevant to the manufacturing industry because in practice many objects are not modeled by polyhedra of arbitrary shape complexity.

5.6.1 Monotone Polyhedra

A polygon P is monotonic in direction l if for every line L orthogonal to l that intersects P, the intersection $L \cap P$ is a line segment (or point). We generalize this notion to 3-dimensions to obtain a *large* family of monotone polyhedra. We define the class as follows.

Definition 5.6.1 *A polyhedron P is* weakly monotonic *in direction l if there exists a direction l such that the intersection, of each plane orthogonal to l that intersects P, is a* simple polygon *(or a line segment or point). The direction l is referred to as the direction of monotonicity.*

Note that there exist many different classes of simple polygons [63], [69], [85]. By substituting one of these classes for the word *simple* in the above definition, we obtain a score of families of *weakly monotonic* polyhedra. Thus we say that if all the intersections are *convex* polygons, we have a weakly monotonic polyhedron in the *convex sense*. If the intersections are *monotone* polygons, then we have a weakly monotonic polyhedron in the *monotone sense*, and so on. Refer to Figure 5.5. Weakly monotone polyhedra have been previously investigated in the context of movable separability of polyhedra [85].

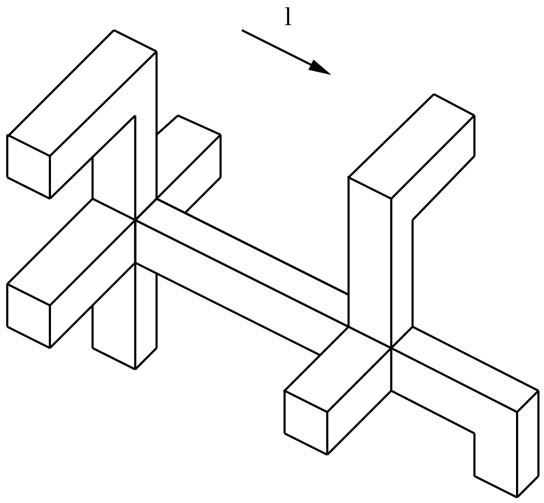

Figure 5.5: Weakly Monotonic Polyhedron

Theorem 5.6.1 *A weakly monotonic polyhedron P is 1-fillable if it is oriented such that gravity points in the direction of monotonicity.*

Proof: For ease of exposition, let us assume that gravity, g, is in the negative z-direction. If we show that P has only one local maximum in the positive z-direction then by theorem 5.3.1 we establish the theorem. Suppose that P had more than one local maximum. Let m be a local maximum that is not the global z-maximum. Let P_m be the union of the facets incident to m, and let h_m be the plane containing m with normal g.

Let h_m^- be the lower closed half-space bounded by the plane h_m with normal g, containing the vertex m. By definition 5.3.2, we have that $P_m \in h_m^-$. Since there is a point with a greater z value than m, the intersection of h_m with P is not a simple polygon, a contradiction. ∎

5.6.2 Facet-Visible Polyhedra and Star-Shaped Polyhedra

Two points inside a polyhedron are said to be *visible* if the line segment between them does not intersect the exterior of the polyhedron. A point p is *weakly visible* from a facet f if there is a point x on f such that p is visible from x.

A polyhedron P is *facet-visible* if there is a facet of the polyhedron from which all the points in the polyhedron are weakly visible. Let P be a facet-visible polyhedron. Without loss of generality, let f_1 be the facet from which the polyhedron is weakly visible. Let d^* denote the direction of the interior normal to the facet.

Theorem 5.6.2 *A facet-visible polyhedron P is 1-fillable if it is oriented such that d^* points in the direction of gravity.*

Proof: For ease of exposition, let us assume that gravity is in the negative z-direction.

Let p_1, an arbitrary point of the facet, be the pin gate. Let a be an arbitrary point in P. Since P is facet-visible, there must be a point b on f_1 that sees point a, i.e. $[ab] \in P$.

Let Π be the path $= (a, b, p_1)$ in P. Since Π is monotone with respect to d^*, the theorem follows. ■

Corollary 5.6.1 *Every polyhedron that is weakly visible from a sectional polygon is 2-fillable with re-orientation.*

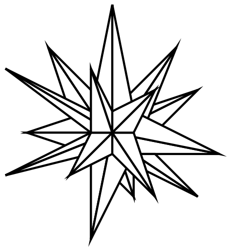

Figure 5.6: A star-shaped polyhedron that is not 1-fillable

A *star-shaped polyhedron* is a polyhedron that contains at least one point x from which all points of the polyhedron are visible (see Figures 5.1 and 5.6 for a star-shaped polyhedron). The set of points from which all points are visible is known as the *kernel* of the star-shaped polyhedron. A point in the kernel of a star-shaped polyhedron can be computed in $O(n)$ time using Megiddo's linear programming technique [57]. This implies that in $O(n)$ time, a sectional polygon can be found from which the star-shaped polyhedron is weakly visible. However, a star-shaped polyhedron may not necessarily be 1-fillable (see Figure 5.6). In fact, if a star-shaped polyhedron is filled from one fixed orientation, it may need $\Omega(n)$ venting holes.

Theorem 5.6.3 *A star-shaped polyhedron is not necessarily 1-fillable but can always be 2-filled with re-orientation in $O(n)$ time.*

5.6.3 Other Restricted Polyhedra

In this subsection, we simply point out that improvements on the $O(n^2)$ time algorithm have been found for polyhedra satisfying certain regularity conditions. These are local conditions imposed on each convex vertex to ensure that the resulting convex polygons that are obtained for the covering problem are *fat* (see [54, 46], that is, the ratio of the diameter of the polygon to its width is bounded by a constant. The only reason algorithm 7 used $O(n^2)$ time due to the fact that the arrangement of convex polygons can have $O(n^2)$ complexity. However, an arrangement of *fat* convex polygons does not have $O(n^2)$ complexity. Therein lies the improvement. A detailed treatment of this topic can be found in [14].

Chapter 6

Stereolithography

6.1 Introduction

In this chapter, we consider the problem of deciding whether or not a design is feasible for a CAD/CAM system developed and patented by 3D Systems of Sylmar, CA that employs a process called *stereolithography* (See Figure 6.1). Stereolithography is emerging as the dominant process used for rapid prototyping. The components of the stereolithography manufacturing process consist of a vat of liquid photocurable plastic, a computer controlled table T on a stand S that can be moved up and down in the vat and a laser L above the vat that can shine on the surface of the liquid plastic and can move in a horizontal plane. The system works as follows. At the first step the table is just below the surface of the plastic and the laser is controlled to move about so that the light shines on the surface of the plastic and *draws* the bottom-most cross-section of the object A being built. When the laser light contacts the plastic, the plastic solidifies and so the first cross-section of the object is formed and rests on the table. At the next step the table is lowered a small amount to allow liquid to cover the hardened layer and the laser then draws the next cross-section of the object. The light from the laser penetrates the liquid just deep enough so that this cross-section is welded to the lower cross-section produced at the previous step. This process is repeated until the entire object is formed. The direction given by a normal to the table pointing from the laser is called the *direction of formation* for the object.

There are some objects that can be formed only if the direction of formation is chosen correctly. For example, in Figure 6.2, the object (a) can not be formed in the position shown. Consider what occurs when the cross-section is reached where the surface S lies. The surface S is not supported below and so as it is formed it sinks to the level of the table. However, if the object is formed in the opposite direction as in Figure 6.2 (b) then stereolithography will succeed. Naturally, there are some objects that can not be formed using stereolithography regardless of the direction of formation chosen.

In order to better understand this manufacturing process, we define a mathematical model of stereolithography (referred to as *vertical stereolithography*). Under this model, we assume that each layer can be welded on to the previous such that no part of the top layer hangs over the previous. We analyze the class of objects that can be constructed under the assumptions of the model. Given an object (modelled as a polygon or a polyhedron), we decide if a direction of formation exists that will result in the successful construction of the object. Such a direction will be called a *valid direction of formation*. We provide an $O(n)$ time algorithm for finding a valid direction of formation where n is the number of vertices of the object. Furthermore, if the object is feasible, we report a description of all the orientations in which the object can be made. We then define a more

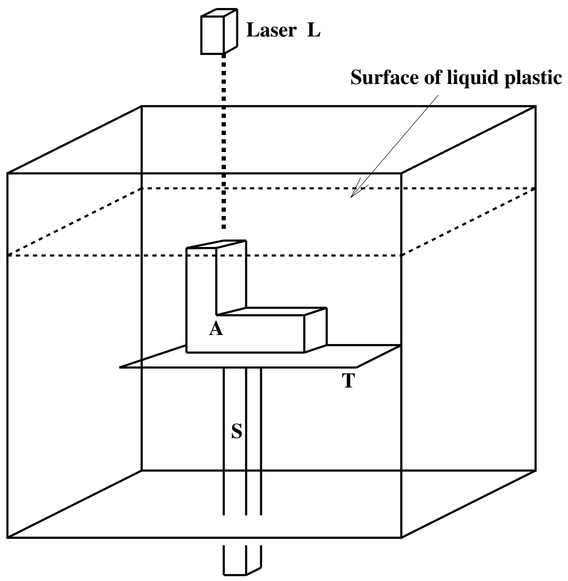

Figure 6.1: Stereolithograpy system.

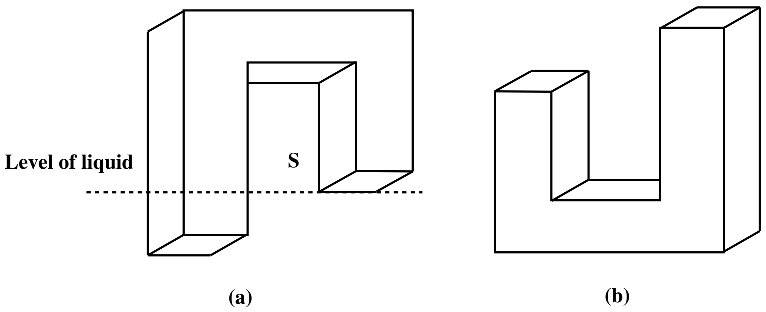

Figure 6.2: Infeasible and feasible directions of formation.

flexible model that more accurately reflects the actual capabilities of stereolithography (referred to as *variable-angle stereolithography*). In this model, we assume that as each layer is welded on to the previous, the top layer may hang over the previous by a certain fixed amount. Again we study the class of feasible objects for this model. We give an $O(n)$ time algorithm for polygons and $O(n \log n)$ as well as $O(n)$ time algorithms for polyhedra.

6.2 Vertical Stereolithography

We first define the geometric model of stereolithography referred to as *vertical stereolithography.*

A polygonal object is assumed to rest on the x-axis and a polyhedral object is assumed to lie on the plane defined by $y = 0$. For a given object A and direction of formation d, let A_d denote the object oriented and positioned according to d. For $y_0 \geq 0$, let $A_d(y_0)$ be the intersection of A_d with the line $y = y_0$ for polygonal objects and the plane $y = y_0$ for polyhedral objects. We refer to $A_d(0)$ as the *base* of the object (with respect to d). A point p of the object with y-coordinate y_0 is said to be *supported* (with respect to a particular direction of formation) if all the points with x (and z) coordinates the same as p and positive y coordinate less than y_0 are in the object. The cross-sections of the object are assumed to be infinitesimally thin and so direction d is a valid direction of formation for an object if the resulting orientation of the object is such that all points in the object are supported. An object is referred to as *feasible* provided it has at least one valid direction of formation.

6.2.1 Polygonal Objects

In this subsection we consider the two-dimensional problem where the object A we wish to form under the vertical stereolithography model is a simple polygon. Let $v_0, v_1, \ldots, v_{n-1}$ be the clockwise ordering of the vertices around A such that each pair of consecutive vertices v_i, v_{i+i} is joined by an edge e_i (all indices are taken modulo n). For $1 \leq i \leq n$, let θ_i be the angle formed by e_{i-1} and e_i in the interior of A. If edge e_i is such that θ_{i+1} and θ_i are both less than or equal to $\pi/2$ then e_i is called an *acute edge.* If e_i is an acute edge and at least one of θ_{i+1} or θ_i is strictly less than $\pi/2$ then e_i is said to be a *strictly acute edge.* Let n_i denote the direction normal to edge e_i pointing out of the polygon. Let $\mathcal{N}$ be the set of all outer normals.

We first observe a simple geometric fact that will be useful in establishing many of the lemmas and theorems to follow. Let e_i be an edge of polygon A. Let p be a point on the open edge e_i. Let r be a ray emanating from point p in direction d.

Observation 6.2.1 *There exists a point $q \in r$ distinct from p such that (pq) is contained in ext(A) if and only if $d \cdot n_i$ is positive (i.e. the angle between d and n_i is strictly less than $\pi/2$).*

We begin by showing that the base of a feasible object must be an edge.

Lemma 6.2.1 *If d is a valid direction of formation for polygon A, then $A_d(0)$ is some edge of A.*

Proof: If $A_d(0)$ is not an edge, then it must be a vertex, say v_i. Since both v_{i-1} and v_{i+1} are above the line $y = 0$, at least one of the two cannot be supported by Observation 6.2.1. ∎

The above lemma restricts our search for a valid direction of formation to the outer normals of the edges of a polygon, namely the set $\mathcal{N}$. Therefore, edge e_i of polygon A is said to be a *valid base* if n_i is a valid direction of formation. A point p in A_{n_i} is said to be *vertically visible* from e_i if the vertical line segment from p to e_i is contained in A_{n_i}. Thus, we observe the following

Observation 6.2.2 *A polygon A_{n_i} is feasible with valid base e_i if and only if all points in A_{n_i} are vertically visible from e_i.*

Although Observation 6.2.2 provides some insight into the structure of a feasible polygon, the following characterization of feasible polygons is useful from a computational perspective.

Lemma 6.2.2 *An edge e_i of A_{n_i} is a valid base if and only if $n_i \cdot n_j \leq 0$ ($\forall\ 1 \leq j \leq n,\ j \neq i$).*

Proof:
($\Rightarrow$) Suppose e_i is a valid base but there exists an edge e_j such that $n_i \cdot n_j > 0$. Consider a point p on the open edge e_j. Let q be the orthogonal projection of p onto the line $L(e_i)$. The open line segment (pq) must be contained in A_{n_i}. However, this is impossible by Observation 6.2.1.

($\Leftarrow$) Suppose that $n_i \cdot n_j \leq 0$ ($\forall\ 1 \leq j \leq n,\ j \neq i$), but e_i is not a valid base. Then there must exist some point p in A_{n_i} that is not vertically visible from e_i by Observation 6.2.2. Let q be the orthogonal projection of p onto $L(e_i)$. Line segment $[pq]$ must intersect $bd(A_{n_i})$ above $L(e_i)$ since p is not vertically visible from e_i. Let x be the intersection point of $[pq]$ and $bd(A_{n_i})$ closest to p. Let us assume for the moment that x is on the open edge e_k. Line segment $[px]$ must be in A_{n_i} since p is in A_{n_i} and x is the first intersection with the boundary. Let y be the intersection of $[xq]$ with $bd(A_{n_i})$ closest to x or q if no such intersection exists. Line segment (xy) is contained in $ext(A_{n_i})$. But this implies that $n_k \cdot n_i > 0$ by Observation 6.2.1 which is a contradiction. A similar argument holds had x been a vertex. ■

With this in mind, we uncover a key characteristic of valid bases, that leads to a linear time algorithm.

Lemma 6.2.3 *If e_i is a valid base then e_i is acute.*

Proof: Suppose e_i is a valid base that is not acute. Then either $n_{i-1} \cdot n_i > 0$ or $n_{i+1} \cdot n_i > 0$ or both. By Lemma 6.2.2 this contradicts the fact that e_i is valid. ■

Given this characteristic, we completely characterize the convex objects that are feasible. The following lemma shows that for a convex object A there is a simple linear time test to find a valid base for A or report that none exists.

Lemma 6.2.4 *Given a convex polygon A, the edge e_i is a valid base if and only if e_i is acute.*

Proof:
($\Rightarrow$) If e_i is a valid base, then by Lemma 6.2.3 it must be acute.

($\Leftarrow$) Since e_i is acute, extending e_{i-1} and e_{i+1} causes them to meet at a point directly above some point of e_i, thus forming a triangle with e_i that is vertically visible from e_i. By convexity, A must lie in this triangle and so for any point p in A there is a point q on e_i vertically below p. Therefore, by Observation 6.2.2, e_i is a valid base. ■

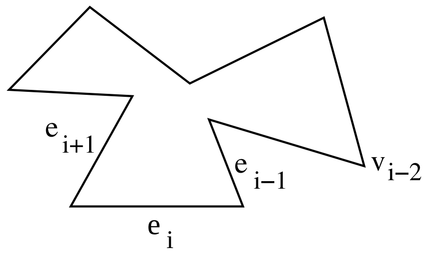

Figure 6.3: A non-convex object with an acute edge that is not a valid base.

The characterization of convex objects in Lemma 6.2.4 implies that a simple examination of the angles between the edges of a convex object is sufficient to find a valid base if one exists or report that the object is not feasible. For a non-convex object, such local tests on the angles are insufficient to determine the feasibility of an object, since such an object may have an acute edge that is not a valid base. For example, in Figure 6.3, edge e_i is an acute edge but not a valid base of the polygon since vertex v_{i-2} is not supported. However, the following lemma shows the relationship between the feasibility of a simple polygon and its convex hull.

Lemma 6.2.5 *If simple polygon A_{n_i} is feasible with base e_i then the convex hull of A_{n_i} is also feasible with base e_i.*

Proof: Follows from Observation 6.2.2, Lemma 6.2.3 and Lemma 6.2.4. ■

Since the convex hull of a simple polygon can be computed in linear time ([56], [59]) and a convex polygon can only have at most 4 acute edges, we see that feasibility of a simple polygon can be computed in linear time. The convex hull of a simple polyhedron, however, cannot be computed in linear time, but can be computed in $O(n \log n)$ time (see [69]). Therefore, although this approach provides an optimal solution to the problem in two dimensions, a solution in three dimensions will require an additional $\log n$ factor. To this end, we explore the following alternate solution that can be generalized to the three-dimensional version of the problem.

Let us first examine the restrictions that the existence of a strictly acute edge puts on the feasibility of a non-convex polygon.

Lemma 6.2.6 *If a simple polygon A is feasible and edge e_i of A is strictly acute then the set of all valid bases of A is a non-empty subset of $\{e_i, e_{i-1}, e_{i+1}\}$.*

Proof: (Refer to Figure 6.3). Suppose that none of e_i, e_{i-1} and e_{i+1} are valid. Since e_i is strictly acute, without loss of generality, assume that $\theta_i < \pi/2$. Since A is feasible, let e_j be a valid base of A. Notice that n_j cannot be contained in $NH(n_i)$ since otherwise $n_j \cdot n_i > 0$. Similarly, n_j cannot be in $arc[N^+(n_i), opp(n_i))$ because otherwise $n_j \cdot n_{i+1} > 0$. Also, n_j cannot be in $arc[opp(n_i), N^-(n_i)]$ since otherwise $n_j \cdot n_{i-1} > 0$. But $NH(n_i) \cup arc[N^+(n_i), opp(n_i)) \cup arc[opp(n_i), N^-(n_i)]$ represents all directions. Therefore, n_j cannot exist. ■

Lemma 6.2.3 guarantees that an acute edge e_i exists if A is feasible and Lemma 6.2.6 says that if a strictly acute edge e_i exists then it is sufficient to test e_i, e_{i-1} and e_{i+1} for a valid base. We now consider what happens when e_i is an acute edge with both θ_{i+1} and θ_i equal to $\pi/2$. If A_{n_i} contains a unique edge e_j such that n_j is $opp(n_i)$ then we label the edge $e_{top}(i)$.

Lemma 6.2.7 *If A is feasible and e_i is an acute edge such that $\theta_{i+1} = \theta_i = \pi/2$ then the set of all valid bases of A is a non-empty subset of $\{e_i,\ e_{i-1},\ e_{i+1}, e_{top}(i)$ (if it exists) $\}$.*

Proof: Similar to the proof of Lemma 6.2.6. ∎

With Lemma 6.2.7 we have characterized all polygons that are feasible. We summarize with the following theorem.

Theorem 6.2.1 *Given that A contains an acute edge e_i, the set of all valid bases of A is a non-empty subset of $\{e_i,\ e_{i-1},\ e_{i+1}, e_{top}(i)$ (if it exists and $\theta_{i+1} = \theta_i = \pi/2$) $\}$ if and only if polygon A is feasible.*

Proof: Follows from Lemma 6.2.3, Lemma 6.2.6, and Lemma 6.2.7. ∎

Determining whether or not a polygon has an acute edge can be achieved in $O(n)$ time, where n is the number of vertices of the polygon. Thus, in $O(n)$ time, the number of possible valid bases can be reduced to 3 or 4 by Theorem 6.2.1. Moreover, by Lemma 6.2.2 we can test in $O(n)$ time whether any of these candidate edges is valid simply by testing its outward normal with the outward normals of all the other edges. Therefore, we can test a polygon A for feasibility and find all valid bases in $O(n)$ time.

Theorem 6.2.2 *In $O(n)$ time the feasibility of a polygonal object with n vertices can be determined and all valid bases identified when the object is feasible.*

6.2.2 Polyhedral Objects

In this subsection we consider the three-dimensional case where the object is a simple polyhedron. We want to find a facet of polyhedron A that is a valid base or determine that A is not feasible.

The following notation will be used in this subsection. Let A be a polyhedron with n vertices. Given a facet f of a polyhedron, we denote the plane containing f by $P(f)$. For facet f of A, let $f(1), f(2), \ldots, f(k_f)$ be the facets of A that share at least one edge with f. Let $\theta_i(f)$ be the angle interior to A between the plane $P(f)$ and the plane $P(f(i))$ about the line of intersection of $P(f)$ and $P(f(i))$. If $\theta_i(f) \leq \pi/2$ for all i, $1 \leq i \leq k_f$, then f is called an *acute facet.* If f is acute and for some i, $\theta_i(f) < \pi/2$, then f is said to be a *strictly acute facet.* Let $n(f)$ denote the direction normal to facet f pointing out of the polyhedron. Let $\mathcal{N}$ be the set of all outer normals. We show several properties analogous to those in the previous subsection that will give rise to a linear time feasibility testing algorithm. We first observe a simple geometric fact. Let f be a facet of polyhedron A. Let p be a point on the facet f. Let r be a ray emanating from point p in direction d.

Observation 6.2.3 *There exists a point $q \in r$ distinct from p such that (pq) is contained in $ext(A)$ if and only if $d \cdot n(f)$ is positive (i.e. the angle between d and $n(f)$ is strictly less than $\pi/2$).*

We begin by showing that the base of a feasible object must be a facet.

Lemma 6.2.8 *If d is a valid direction of formation for polyhedron A, then $A_d(0)$ is some facet of A.*

Proof: If $A_d(0)$ is not a facet, then it must either be an edge or a vertex. If it is an edge e, then let f_i and f_j be the two facets adjacent to e. Since both facets lie above a plane containing e, either $n(f_i) \cdot d$ or $n(f_j) \cdot d$ is positive. Without loss of generality, assume it to be $n(f_i)$. By Observation

6.2.3 there is a point on the facet f_i that is not supported. A similar argument holds if $A_d(0)$ is a vertex. ∎

The above lemma restricts our search for a valid direction of formation to the outer normals of the facets of a polyhedron, namely the set $\mathcal{N}$. Therefore, facet f of polyhedron A is said to be a *valid base* provided that $n(f)$ is a valid direction of formation. A point p in $A_{n(f)}$ is said to be *vertically visible* from f if the vertical line segment from p to f is contained in $A_{n(f)}$. Thus, we observe the following.

Observation 6.2.4 *A polyhedron $A_{n(f)}$ is feasible with valid base f if and only if all points in $A_{n(f)}$ are vertically visible from f.*

As in the two dimensional case, the following characterization of feasible polyhedra will prove to be more useful from a computational perspective.

Lemma 6.2.9 *A facet f_i of $A_{n(f_i)}$ is a valid base if and only if $n(f_i) \cdot n(f_j) \leq 0$* (for all facets f_j of $A_{n(f_i)}$, where $f_j \neq f_i$).

Proof:
($\Rightarrow$) Suppose f_i is a valid base but there exists a facet f_j such that $n(f_i) \cdot n(f_j) > 0$. Consider a point p on the facet f_j. Let q be the orthogonal projection of p onto the plane $P(f_i)$. The line segment $[pq]$ must be contained in $A_{n(f_i)}$. However, this is impossible by Observation 6.2.3.

($\Leftarrow$) Suppose that $n(f_i) \cdot n(f_j) \leq 0$ for all facets f_j of $A_{n(f_i)}$ distinct from f_i, but f_i is not a valid base. Then there must exist some point p in $A_{n(f_i)}$ that is not vertically visible from f_i by Observation 6.2.4. Let q be the orthogonal projection of p onto $P(f_i)$. Line segment $[pq]$ must intersect $bd(A_{n(f_i)})$ above $P(f_i)$ since p is not vertically visible from f_i. Let x be the intersection point of $[pq]$ and $bd(A_{n(f_i)})$ closest to p. Let us assume for the moment that x is on the facet f_j. Line segment $[px]$ must be in $A_{n(f_i)}$ since p is in $A_{n(f_i)}$ and x is the first intersection with the boundary. Let y be the intersection of $[xq]$ with $bd(A_{n(f_i)})$ closest to x or q if no such intersection exists. Line segment (xy) is contained in $ext(A_{n(f_i)})$. But this implies that $n(f_j) \cdot n(f_i) > 0$ by Observation 6.2.3 which is a contradiction. A similar argument holds for the case where x is a vertex or on an edge. ∎

Lemma 6.2.10 *If facet f is a valid base for polyhedron A then f is acute.*

Proof: Suppose that f is a valid base for A but f is not acute. Then there must be some $f(i)$ such that $\theta_i(f) > \pi/2$. However, this implies that $n(f) \cdot n(f(i)) > 0$. By Lemma 6.2.9, this contradicts the fact that f is valid. ∎

In the special case of convex polyhedra, we see that a simple local test on each facet suffices to determine if a facet is a valid base.

Lemma 6.2.11 *Let A be a convex polyhedron. Face f is a valid base if and only if f is acute.*

Proof: Similar to proof of Lemma 6.2.4. ∎

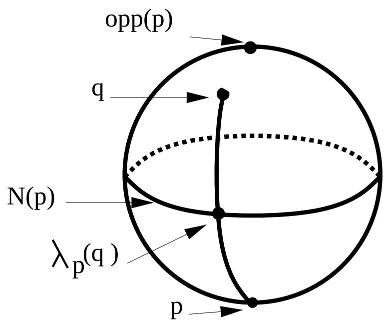

Figure 6.4: Illustrating $\lambda_p(q)$

It is no longer clear whether the feasibility of a convex polyhedron can be determined in $O(n)$ time since a facet f of a polyhedron may have $O(n)$ adjacent facets. However, the total complexity of all adjacencies is linear by Euler's formula (see [11]). Therefore, testing all facets for validity by the local test implied in Lemma 6.2.11 can be done in $O(n)$ time. We now turn our attention to polyhedral objects that are not necessarily convex. The following lemma shows the relationship between the feasibility of a simple polyhedron and its convex hull.

Lemma 6.2.12 *If simple polyhedron $A_{n(f)}$ is feasible with base f then the convex hull of $A_{n(f)}$ is also feasible with base f.*

Proof: Follows from Observation 6.2.4, Lemma 6.2.10, and Lemma 6.2.11. ∎

Lemma 6.2.12 implies the following simple approach to determine if a given polyhedron A is feasible. Compute the convex hull of A in $O(n \log n)$ time. A convex polyhedron can have at most 6 acute facets. Each acute facet of the convex hull is a candidate base. Testing a facet can be done in linear time by Lemma 6.2.9. Therefore, determining feasibility of a simple polyhedron can be achieved in $O(n \log n)$ time. The complexity is dominated by the computation of the convex hull. To circumvent the computation of the convex hull, we explore the following approach which will lead to an optimal algorithm.

We first examine the restrictions placed on the feasibility of a polyhedron in the presence of a strictly acute facet. Before doing so, we define the following geometric term (see Figure 6.4). Let p be a point on the sphere of directions $\mathcal{S}$. Let q be any point on $\mathcal{S}$ distinct from p and $opp(p)$. We define $\lambda_p(q)$ to be the point on $N(p)$ closest to q (i.e. the intersection point closest to q of $N(p)$ with the great circle through p and q).

We show that if the polyhedral object A has a strictly acute facet f, then f or one of its adjacent facets must be a valid base if the object is feasible.

Lemma 6.2.13 *If polyhedron A is feasible and f is a strictly acute facet then the set of all valid bases of A is a non-empty subset of $\{f, f(1), \ldots, f(k_f)\}$.*

Proof: Suppose that none of f, $f(1), \ldots$ and $f(k_f)$ are valid. Since f is strictly acute, without loss of generality, assume that $\theta_i(f) < \pi/2$. Since A is feasible, let f_j be a valid base of A. We see that $n(f_j) \neq opp(n(f))$ since $n(f(i)) \cdot opp(n(f)) > 0$. This implies that $\lambda_{n(f)}(n(f_j))$ is properly defined. Now, we know that $n(f_j)$ cannot be in $NH(n(f))$ for this would violate the validity of facet f_j by Lemma 6.2.9. Therefore, $n(f_j)$ must be in $NH^c[n(f)]$.

We notice that $\lambda_{n(f)}(n(f(i))$ is simply the outward normal of the edge of facet f (which is a polygon) corresponding to the intersection of $f(i)$ and f. It follows that every open half-circle $\subset N(n(f))$ contains at least one point of $\lambda_{n(f)}(f(1)), \lambda_{n(f)}(f(2)), \ldots,$ or $\lambda_{n(f)}(f(k_f))$ since facet f is a simple polygon. Therefore, given a point $x \neq opp(n(f))$ in $NH^c[n(f)]$, there exists a facet $f(i)$ adjacent to f such that $\lambda_{n(f)}(x) \cdot \lambda_{n(f)}(n(f_i)) > 0$. Observe, however, that if two directions $a, b \in NH^c[n(f)]$ both distinct from $opp(n(f))$ are such that $\lambda_{n(f)}(a) \cdot \lambda_{n(f)}(b) > 0$, then $a \cdot b > 0$. But this implies that $n(f_j)$ cannot exist. ■

If f is not strictly acute, we define f_{top} analogously to $e_{top}(i)$ in the previous subsection. We have the following lemma.

Lemma 6.2.14 *If polyhedron A is feasible and f is an acute, but not strictly acute, facet then the set of all valid bases of A is a non-empty subset of $\{f, f(1), \ldots, f(k_f), f_{top}$ (if it exists) $\}$.*

Proof: Similar to the argument given in the proof of Lemma 6.2.13. ■

These results were sufficient in the two-dimensional case to reduce the number of candidate bases to at most 4. Unfortunately, in the 3-dimensional case, an acute facet f may have $O(n)$ adjacent facets. However, we are able to link the feasibility of a facet in a polyhedron to the feasibility of an edge in a polygon. Thus, we establish the following theorem.

Theorem 6.2.3 *Given that A has an acute facet f, polyhedron A is feasible if and only if the set of all valid bases of A is a non-empty subset of $\{f$, f_{top} (if it exists) and at most 4 facets adjacent to $f\}$. Moreover, the edges corresponding to the intersection of f with the at most 4 facets adjacent to f are valid edges for polygon f.*

Proof:

($\Rightarrow$) If the set of valid bases of A is a non-empty subset of: f, f_{top} (if it exists) and at most 4 facets adjacent to f, then by definition, A is feasible.

($\Leftarrow$) If A is feasible, we must show that the following facets of A are the only valid bases: f, f_{top} (if it exists) and at most 4 facets adjacent to f. Lemma 6.2.14 reduces our task to showing that at most 4 facets adjacent to f can be bases. Suppose 5 facets adjacent to f were valid bases. Let us denote them by $f(i_1), f(i_2), \ldots, f(i_5)$. Notice that $n(f(i_1))$, $n(f(i_2))$, $n(f(i_3))$, $n(f(i_4))$, and $n(f(i_5))$ are all contained in $NH^c[n(f)]$ since f is acute. Also, since they are all valid bases, $n(f(i_j)) \cdot n(f_k) \leq 0$ for all $1 \leq j \leq 5$ and for all facets $f_k \neq f(i_j)$ of A by Lemma 6.2.9.

Let $f(1), f(2), \ldots, f(k_f)$ be the facets adjacent to facet f. Since f is acute $n(f(1)), \ldots, n(f(k_f))$ are all contained in $NH^c[n(f)]$. Observe that $\lambda_{n(f)}(n(f(k))$ is properly defined for all $1 \leq k \leq k_f$. Since each of $f(i_1), f(i_2), \ldots, f(i_5)$ is a valid base, we have that $\lambda_{n(f)}(n(f(i_j))) \cdot \lambda_{n(f)}(n(f(k)) \leq 0$ for all $1 \leq j \leq 5$ and all facets $f(k)$ adjacent to f distinct from $f(i_j)$. We notice that $\lambda_{n(f)}(n(f(k))$ is simply the outward normal of the edge of facet f (which is a polygon) corresponding to the intersection of $f(k)$ and f. But this would mean that polygon f has 5 valid edges by Lemma 6.2.2, contradicting Theorem 6.2.1. ■

Therefore, the number of possible valid bases in a feasible polyhedron A is at most 6. We summarize below the linear time algorithm to determine the feasibility of a simple polyhedron. The algorithm takes a simple polyhedron A as input.

Algorithm 8: *Determine the feasibility of a simple polyhedron.*

1. Determine if A has an acute facet. If A does not have an acute facet, exit (A is not feasible).

2. Let f be the acute facet of A. Scan all other facets of A to determine if f_{top} exists.
3. Compute all possible valid edges of polygon f using the algorithm described in Section 6.2.1. There are at most 4 edges. Let F' represent the facets of A adjacent to these edges excluding facet f.
4. Let B be $\{f, f_{top}$ (if it exists) $\} \cup F'$. The set B represents the candidate bases of A. There are at most 6 facets in B by Theorem 6.2.3.
5. Test each facet $f_i \in B$ to see if it is valid in the following way:
   Check that the angle between normal $n(f_i)$ and all other normals is no less than $\pi/2$. This can be done in linear time.
6. Output the valid bases.

The correctness of the algorithm follows from Theorem 6.2.3. As for the time complexity, we see that step 1 can be done in O(n) time by Euler's formula (see [11]). Step 3 takes linear time by the algorithm given in Section 6.2.1. Furthermore, by Lemma 6.2.9 testing each candidate facet can be done in $O(n)$ time simply by testing its outward normal with the outward normals of all the other facets. Since there are only a maximum of 6 candidate facets, we conclude that testing a polyhedron A for feasibility and finding all valid bases can be achieved in $O(n)$ time.

Theorem 6.2.4 *In $O(n)$ time the feasibility of a polyhedral object with n vertices can be determined and all valid bases identified when the object is feasible.*

6.2.3 Relation to NC machining

A 3-axis NC machine consists of a worktable, a spindle or milling cutter, and the motors and controls for positioning the cutter and/or the worktable along the three translational axes corresponding to the three axes of a Cartesian coordinate system (see Held [42] for a discussion on the different types of NC machines). A cutter can be viewed as a thin cylinder or rod rotating around its axis of symmetry. Without loss of generality, assume this axis of symmetry of the cutter is parallel to the z-axis, and that the object contacts the worktable on a face. Then, any polyhedron P constructed by a 3-axis NC machine has the following property: for every point p on the surface of P (except for the base), there exists a ray emanating from p parallel to the z-axis that does not intersect any other point on P. This follows from the fact that the cutter must reach the point and its movement is restricted to translations along the three coordinate axes. Therefore, we have the following.

Theorem 6.2.5 *A polyhedral object formed by 3-axis NC machining can be recognized in linear time and can also be constructed by vertical stereolithography.*

6.3 Variable-Angle Stereolithography

In practice, as the laser welds one cross-section on to the other, if the top layer is "close enough" to the previous layer, it can be welded on. That is, the upper layer may hang over the previous by a certain amount and still get welded on. To model this mathematically, we define the following model referred to as *variable-angle stereolithography.*

Intuitively, variable-angle stereolithography differs from vertical stereolithography in the following way. As each layer is glued on by the laser, the topmost layer can hang over the previous layer by the freedom allotted by some constant angle ω. More formally, we say that a point p with y-coordinate y_0 is *ω-supported* with respect to the direction of formation if there exists a point q

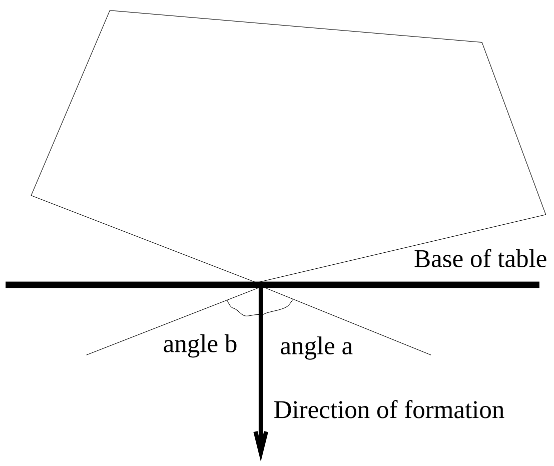

Figure 6.5: A vertex that is a valid base.

with positive y coordinate less than y_0 such that the line segment $[pq]$ is contained in the object and the smaller angle between the direction of formation and the vector $\overrightarrow{pq}$ is less than or equal to ω. Clearly, ω must be less than $\pi/2$. Notice that variable-angle stereolithography is a generalization of vertical stereolithography. The two are equivalent when ω is zero. An object can be built with respect to the parameter ω if there exists an orientation of the object such that all points above the base are ω-supported. An object that can be built with respect to the parameter ω will be called *ω-feasible*.

6.3.1 Polygonal Objects

The parameter ω enlarges the class of objects that can be formed. In fact, with $\omega > 0$, the base of an object no longer need be a edge of the polygon. For example, the polygon in Figure 6.5 is feasible (as long as both $\angle a$ and $\angle b$ are both less than or equal to ω) with a vertex as base. For polygonal objects, we will assume that the base of an object is always an edge, since building an object on a vertex is unstable.

We say that a point p in A_{n_i} is *ω-visible* from e_i if p is above $L(e_i)$ and there exists a polygonal path Π from p to e_i such that $\Pi \in A_{n_i}$ and every vertex in Π (except for the vertex on e_i) is ω-supported by an adjacent vertex. Thus, we observe the following.

Observation 6.3.1 *A polygon A_{n_i} is ω-feasible with valid base e_i if and only if all points in A_{n_i} are ω-visible from e_i.*

A polygonal chain is said to be *monotonic* with respect to direction Θ if the intersection of every line parallel to $N(\Theta)$ with the chain is either empty or a point. We observe the following property that is crucial to the development of a linear algorithm.

Observation 6.3.2 *If a point p is ω-visible from e_i, then there exists a path Π from p to e_i that is* monotone *with respect to direction n_i.*

We present an alternate characterization of ω-feasibility that will be useful from a computational perspective.

Theorem 6.3.1 *A polygon A_{n_i} is ω-feasible with valid base e_i if and only if the angle between n_i and all other normals is no less than $\pi/2 - \omega$ and the set of all local minima with respect to n_i is e_i.*

Proof:
($\Leftarrow$) Given that all n_j distinct from n_i are such that the smaller angle between $\angle n_i n_j \geq \pi/2 - \omega$, and all local minima are contained in e_i, we will show that A_{n_i} is ω-feasible with base e_i. We do this by showing that every point in A_{n_i} is ω-supported by the following construction.

Let $p \in A_{n_i}$. Assume that $p \notin e_i$.

1. If p is contained in $int(A_{n_i})$, then let q be the intersection point below p and closest to p of a vertical line through p and $bd(A_{n_i})$.

2. If p is contained in the interior of an edge e, then let q be the vertex adjacent to e with lower y-coordinate. Such a vertex must exist since p is not a local minimum.

3. If p is a vertex v, then let q be the vertex adjacent to v with lower y-coordinate. Such a vertex must exist since p is not a local minimum.

By construction, the smaller angle between $\overrightarrow{pq}$ and n_i is no more than ω. Therefore, p is ω-supported. If $q \in e_i$, then we are done. If $q \notin e_i$, we must show that q is ω-supported. This can be done by repeating steps 1, 2, 3 with q. The construction must end with a point on e_i since e_i contains all local minima with respect to n_i and with every iteration, the y-coordinate of the newly constructed point is decreased.

($\Rightarrow$) Given that A_{n_i} is ω-feasible with valid base e_i, we will show that the smaller angle between n_i and all other outer normals is greater than or equal to $\pi/2 - \omega$ and that the set of all local minima with respect to n_i is e_i.

Suppose there exists an outer normal n_j such that $\angle n_i n_j < \pi/2 - \omega$. Let p be a point in the interior of e_j. Since e_i is an ω-feasible base, there must exist a point q such that p is ω-supported by q. However such a q does not exist because of n_j.

Similarly, suppose there exists a local minimum point p that is not contained in e_i. Again, the point p is not ω-supported. ■

Theorem 6.3.2 *For fixed ω, a polygon has a constant number of candidate edges that can be valid bases. These candidate edges can be obtained in $O(n)$ time.*

Proof: Let $k = \lceil 2\pi/(\pi/2 - \omega) \rceil$. Cover the circle of directions with k closed arcs, denoted by $a_1, a_2, \ldots a_k$, having the following property. The angle spanned by each of the arcs is exactly $(\pi/2 - \omega)$.

For edge e_i, suppose that n_i is contained in the open arc a_j. If edge e_i is a valid base, then by Theorem 6.3.1 there are no other outer normals in the open arc a_j. If n_i had been on the end of the closed arc a_j, then there can be at most one other normal on the other end of closed arc a_j. Therefore, each closed arc can contain the outer normal of at most 2 valid bases. Since there are k arcs, there can be at most $2k$ valid bases. But k is a constant when ω is fixed; therefore, there are only a constant number of valid bases.

The algorithm for obtaining the valid bases follows from the discussion above. ■

We now have all the tools needed to determine the ω-feasibility of a simple polygon in linear time. A brief outline of the algorithm follows. The algorithm takes as input a simple polygon A and parameter ω.

Algorithm 9: *Determine the ω-feasibility of a simple polygon.*

1. Let B represent the set of candidate bases of A. There are only a constant number of edges in B and they can be computed in linear time using the technique described in Theorem 6.3.2.
2. Test each edge $e_i \in B$ to see if it is valid in the following way.
   - Check that the angle between normal n_i and all other normals is no less than $\pi/2 - \omega$. This can be done in linear time.
   - Verify that the set of all local minima with respect to n_i is e_i. This can be done in linear time using the algorithm described in Chapter 4 which determine given a polygon, a specified edge, and a direction, whether the edge is the set of all local minima with respect to the given direction.
3. Output the valid bases

Testing an edge to see if it is valid takes linear time. However, since the number of edges tested is constant, step 2 is completed in linear time. The complexity of the algorithm is linear in the size of the input since the time to complete each step is at most linear. The correctness of the algorithm follows from Theorems 6.3.1 and 6.3.2.

Theorem 6.3.3 *The feasibility of a simple polygon in variable-angle stereolithography can be determined in $O(n)$ time.*

Remark: The technique used to determine the feasibility of a simple polygon with $\omega = 0$ provides an alternate linear time method to compute the feasibility in vertical stereolithography.

6.3.2 Polyhedral Objects

Similar to the two-dimensional case, with $\omega > 0$, the base of an object no longer need be a facet of the polyhedron (see Figure 6.5). However, we will assume that the base of an object is always a facet of the polyhedron, since building an object on a vertex or an edge is unstable.

We say that a point p in $A_{n(f)}$ is *ω-visible* from a facet f if p is above the plane $P(f)$ and there exists a polygonal path Π from p to f such that $\Pi \in A_{n(f)}$ and the smaller angle between every pair of edges in Π is no more than ω. Thus, we observe the following.

Observation 6.3.3 *A polyhedron $A_{n(f)}$ is ω-feasible with valid base f if and only if all points in $A_{n(f)}$ are ω-visible from f.*

We observe another property that is crucial to the development of a linear algorithm.

Observation 6.3.4 *If a point p is ω-visible from f, then the path Π from p to f is* monotone *with respect to direction $n(f)$.*

Theorem 6.3.4 *A polyhedron $A_{n(f)}$ is ω-feasible with valid base f if and only if the angle between $n(f)$ and all other normals is no less than $\pi/2 - \omega$ and the set of all local minima with respect to $n(f)$ consists of facet f.*

Proof:

($\Leftarrow$) Given that all outer normals $n(f_j)$ distinct from $n(f)$ are such that the smaller angle between $\angle n(f_j)n(f) \geq \pi/2 - \omega$, and all local minima are contained in f, we will show that $A_{n(f)}$ is ω-feasible with base f. We do this by exhibiting a construction such that every point in $A_{n(f)}$ is ω-supported by the following construction.

Let $p \in A_{n(f)}$. Assume that $p \notin f$.

1. If p is contained in $int(A_{n(f)})$, then let q be the intersection point below p and closest to p of a vertical line through p and $bd(A_{n(f)})$.

2. If p is contained in the interior of a facet f_j, then let q be a point on f_j with lowest y-coordinate.

3. If p is contained in the interior of an edge e, then let q be a point with lowest y-coordinate in one of the two facets adjacent to e.

4. If p is a vertex v, then let q be a point with lowest y-coordinate in one of the facets adjacent to v.

In all cases, q will have a lower y-coordinate than p since p is not a local minimum. By construction, the smaller angle between $\overrightarrow{pq}$ and $n(f)$ is no more than ω. Therefore, p is ω-supported. If $q \in f$, then we are done. If $q \notin f$, we must show that q is ω-supported. This can be done by repeating steps 1, 2, 3, 4 with q. The construction must end with a point on f since f contains all local minima with respect to $n(f)$ and with every iteration, the y-coordinate of the newly constructed point is decreased.

($\Rightarrow$) Given that $A_{n(f)}$ is ω-feasible with valid base f, we will show that the smaller angle between $n(f)$ and all other outer normals is greater than or equal to $\pi/2 - \omega$ and that the set of all local minima with respect to $n(f)$ is e_i.

Suppose there exists an outer normal $n(f_j)$ such that $\angle n(f_j)n(f) < \pi/2 - \omega$. Let p be a point in the interior of f_j. Since f is an ω-feasible base, there must exist a point q such that p is ω-supported by q. However such a q does not exist because of $n(f_j)$.

Similarly, suppose there exists a local minimum point p that is not contained in f. Again, the point p is not ω-supported.

∎

Theorem 6.3.5 *For fixed ω, a polyhedron has a constant number of candidate facets that can be valid bases. These facets can be obtained in $O(n)$ time.*

Proof: Let us consider the spherical coordinates (ϕ, ρ) of the sphere of directions $\mathcal{S}$ centered at the origin where the angle ϕ is in the set $[0, 2\pi)$ and the angle ρ is in the interval $[-\pi/2, \pi/2]$. We first divide the sphere of directions into $k = \lceil \pi/(\pi/4 - \omega/2) \rceil$ slices with parallel circles in the following way. Slice s_1 contains all points where $\rho \in [\pi/2, \pi/2 - (\pi/4 - \omega/2)]$. Slice s_2 contains all points where $\rho \in [\pi/2 - (\pi/4 - \omega/2), \pi/2 - 2(\pi/4 - \omega/2)]$. Slice s_i contains all points where $\rho \in [\pi/2 - (i-1)(\pi/4 - \omega/2), \pi/2 - i(\pi/4 - \omega/2)]$. See Figure 6.7.

Each slice s_i is further subdivided into $m = \lceil 2\pi/(\pi/4 - \omega/2) \rceil$ pieces in the following way. Piece s_{i1} contains all points where $\phi \in [0, \pi/4 - \omega/2]$ and $\rho \in [\pi/2 - (i-1)(\pi/4 - \omega/2), \pi/2 - i(\pi/4 - \omega/2)]$. Piece s_{i2} contains all points where $\phi \in [\pi/4 - \omega/2, 2(\pi/4 - \omega/2)]$ and $\rho \in [\pi/2 - (i-1)(\pi/4 - \omega/2), \pi/2 - i(\pi/4 - \omega/2)]$. Piece s_{ij} contains all points where $\phi \in [(j-1)(\pi/4 - \omega/2), j(\pi/4 - \omega/2)]$ and $\rho \in [\pi/2 - (i-1)(\pi/4 - \omega/2), \pi/2 - i(\pi/4 - \omega/2)]$.

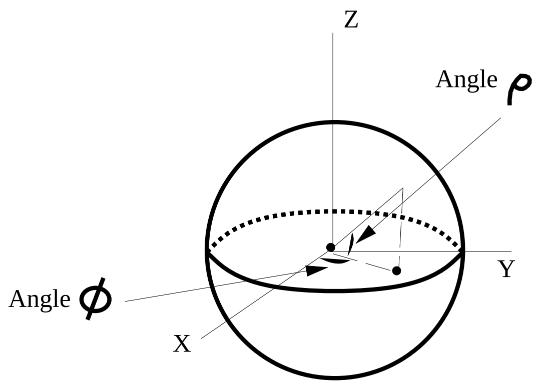

Figure 6.6: Spherical Coordinates.

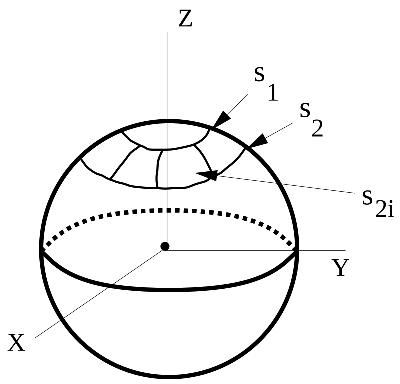

Figure 6.7: Slices and pieces of the sphere of directions.

By construction, any pair of points in a piece s_{ij}, represents a pair of directions d_1 and d_2 such that the smaller angle between d_1 and d_2 is strictly less than $\pi/2-\omega$. Therefore, the outer normals of two feasible bases cannot lie in the same piece.

There are km pieces. Notice that km no more than $\pi^2/(\pi/2-\omega)^2$. Since ω is fixed, $km \in O(1)$. Each piece can contain at most 1 feasible base. Therefore, there are $O(1)$ feasible bases. ∎

We now have all the tools needed to determine the feasibility of a simple polyhedron in linear time. A brief outline of the algorithm follows. The algorithm takes as input a simple polyhedron A and parameter ω.

Algorithm 10: *Determine the ω-feasibility of a simple polyhedron.*

1. Let B represent the set of candidate bases of A. There are only a constant number of facets in B and they can be computed in linear time using the technique described in the proof of Theorem 6.3.5.
2. Test each facet $f_i \in B$ to see if it is valid in the following way.
   - Check that the angle between normal $n(f_i)$ and all other normals is no less than $\pi/2-\omega$. This can be done in linear time.
   - Verify that the set of all local minima with respect to $n(f_i)$ is f_i. This can be done in linear time using the algorithm described in chapter 5 which determines given a polyhedron, a facet and a direction, whether the facet is the set of all local minima with respect to the given direction.
3. Output the valid bases

Testing a facet to see if it is valid takes linear time. However, since the number of facets tested is constant, step 2 is completed in linear time. The complexity of the algorithm is linear in the size of the input since the time to complete each step is at most linear. The correctness of the algorithm follows from Theorem 6.3.4, 6.3.5.

Theorem 6.3.6 *The feasibility of a simple polyhedron in variable-angle stereolithography can be determined in $O(n)$ time.*

Remark: The technique used to determine the feasibility of a simple polyhedron with $\omega = 0$ provides an alternate linear time method to compute the feasibility in standard stereolithography.

The initial assumption that the base of an object is always a facet of the given polyhedron may be slightly weakened at the cost of a $\log n$ factor. One might argue that although the construction of an object from a vertex or edge may be unstable, it is reasonable to assume that the object is placed on a facet of the convex hull of the object. After the construction of the convex hull of the object, we see that determining its feasibility under this weaker assumption can be done in linear time from the discussion above. Therefore, we have the following.

Theorem 6.3.7 *Given a simple polyhedron A, if the base of A can be a facet of its convex hull, then feasibility in variable-angle stereolithography can be determined in $O(n \log n)$ time.*

Chapter 7

Determining if an Object is Castable

7.1 Introduction

In this chapter, we study the problem of determining whether a *re-usable* cast of an object can be constructed. We say that a cast is *re-usable* provided that the cast can be removed from the object without breaking the object or the cast parts. Thus, such a cast can be used more than once in the construction of an object using a casting process. The requirement to remove the cast parts without breaking them, so that they may be re-used, imposes certain restrictions on the shape of the objects that can be constructed. These are the restrictions we investigate in this chapter.

We concentrate on determining if a re-usable two-part cast of an object can be made. Two-part casts are the most popular types of casts used today due to their simplicity and efficiency. To construct a two-part cast, a prototype of the object is first obtained (see Figure 7.1). The prototype is then divided into two parts along a plane. The facet of each prototype part adjacent to the cutting plane is referred to as the base. The first cast part is made by placing the base of the first prototype part on a flat surface, and then adding sand around it. The part is then rotated such that the base is facing up, and the other prototype part is placed such that the bases coincide. The second cast part is built by adding sand around this prototype part while maintaining a channel into the cavity. Once the sand hardens, the cast of the prototype object is complete and the prototype parts can be removed. To build a metal rendition of the prototype object with this cast, liquid metal is poured into the opening until it fills the cavity. After the metal solidifies, the cast parts are removed from the object. The key to constructing a cast with this process is the ability to remove the prototype object without breaking the cast. This property is not restricted to casts built for manufacturing methods related to sand casting but also applies to other metal casting methods [31, 87], as well as injection molding and blow molding methods for plastics [71, 88]. The ability to remove the prototype object from the cast without breaking the cast allows one to re-use the same cast when mass-producing a particular object. Thus for several different manufacturing methods involving casting, the geometry of the object determines its feasibility of construction.

An object is *castable* if it can be manufactured by casting. In other words, a cast of the object can be constructed such that each cast part can be removed from the object without breaking the object or any of the cast parts. Geometric and algorithmic issues of the castability of planar objects have been studied by Rappaport and Rosenbloom [73]. In this chapter, we address casting of objects modelled by polyhedra. In geometric terms, castability can be defined as follows

Definition 7.1.1 *A simple polyhedron P is* castable *if there exists a plane h such that $h^+ \cap \partial P$ is a weak terrain in some orientation, and $h^- \cap \partial P$ is a weak terrain in some orientation. The*

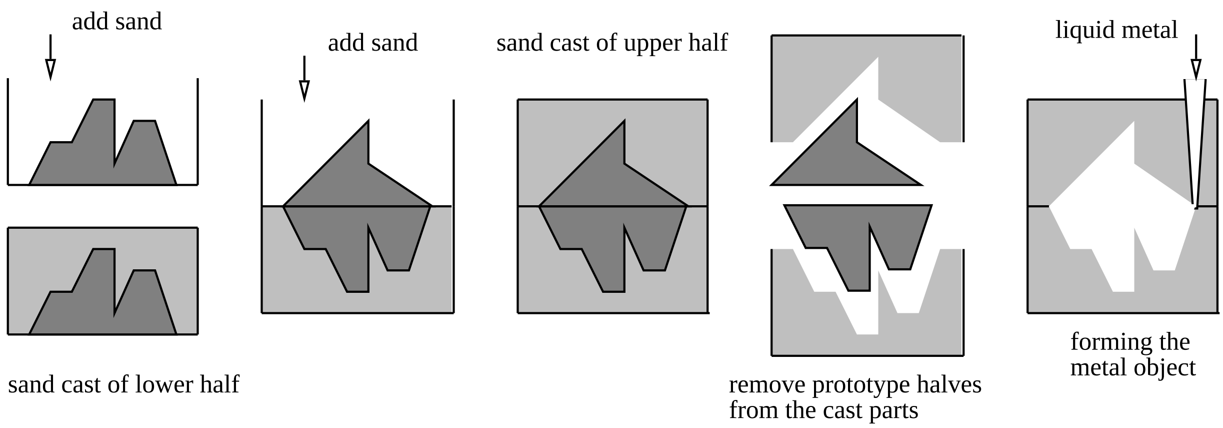

Figure 7.1: Construction of an object by sand casting, using two halves of the object as prototypes.

plane h is called the casting plane*. (A weak terrain may contain edges and facets parallel to the orientation in which it is a terrain.)*

To manufacture a castable object (modelled as a polyhedron P), first determine a casting plane h. The plane h divides P into two cast parts. Make each cast part from the prototype halves $h^+ \cap \partial P$ and $h^- \cap \partial P$. Since P is castable, the prototype halves can be removed from the cast parts, and later the manufactured object can be removed from the cast parts. We consider three versions of the castability problem. They differ in the way the cast parts may be removed from the polyhedron P. Figure 7.2 shows the three versions for planar polygons.

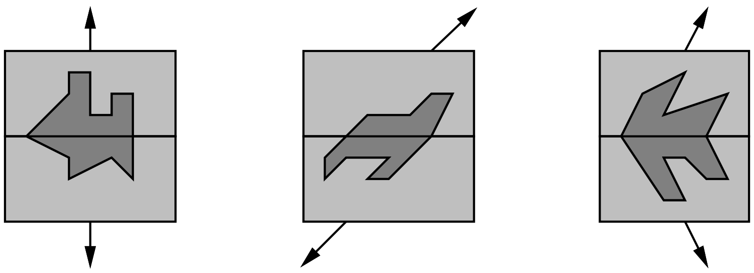

Figure 7.2: Three versions of the castability problem.

1. The two cast parts must be removed from P by one translation each, in opposite directions, and normal to the casting plane (orthogonal cast removal).

2. The two cast parts must be removed from P by one translation each, and in opposite directions (opposite cast removal).

3. The two cast parts must be removed from P by one translation each, in arbitrary directions (arbitrary cast removal).

Any convex polygon (in the plane) is castable in any of the three versions. In three dimensions, the equivalent property does not hold for convex polyhedra; in fact, some convex polyhedra are not castable in any of the three versions. In manufacturing, developing machines that perform orthogonal and opposite cast removal is much simpler than machines that perform arbitrary cast removal. In fact, opposite cast removal seems to be the most popular technique used [24, 71]. Furthermore, if orthogonal or opposite cast removal is possible, it can be determined more efficiently.

7.2 Preliminaries

A polyhedral surface S is called a *weak terrain with respect to a direction* $\vec{d}$ if any line with orientation $\vec{d}$ intersects S in a point or a line segment. A polyhedron P is called a *weak terrain with respect to a facet* Q *and a direction* $\vec{d}$ if $\partial P - Q$ is a weak terrain with respect to $\vec{d}$. In the rest of this paper we use *terrain* to mean *weak terrain*

For an arbitrary plane h, we use h_0^+ and h_0^- to denote h^+ and h^- translated so that the bounding plane intersects the origin. Given direction $\vec{d}$ and facet f, we say that f is *compatible* with $\vec{d}$ if the inner product between $\vec{d}$ and the outward normal of facet f is non-negative (i.e. $\vec{d}$ makes an angle of at most $\pi/2$ radians with the outward normal of f). We say that f is *incompatible* with $\vec{d}$ if it is not compatible.

Observation 7.2.1 *Let P be a polyhedron and let h be a plane that intersects P. The surface $\partial P \cap cl(h^+)$ is a terrain for direction $\vec{d}$ if and only if every facet of P that intersects h^+ is compatible with $\vec{d}$.*

Therefore, castability with respect to a plane h is only determined by the facets of P that intersect h^+ and the ones that intersect h^-. If h is a casting plane for P, then h can be perturbed if this does not involve new facets intersecting h. In case of orthogonal cast removal, the only perturbation allowed is translation.

Observation 7.2.2 *For castability with orthogonal cast removal, we may assume that the casting plane contains at least one vertex of P. For opposite and arbitrary cast removal, we may assume that the casting plane contains at least three vertices of P.*

7.2.1 The sphere of directions

Recall that we represent the space of all directions in 3-space by the points on the surface of a sphere. Let *north* and *south* denote the points on $\mathcal{S}$ that represent the $\vec{z}$ and $-\vec{z}$ directions. Let $\mathcal{E}$ denote the equator (the set of points $p \in \mathcal{S}$, such that $\overrightarrow{op} \cdot \vec{z} = 0$).

Let P be a convex polyhedron, let h be a casting plane and let $\vec{d_1}$ and $\vec{d_2}$ be the two cast removal directions, represented by points d_1 and d_2 on the sphere of directions. We re-orient P and h such that *north* is normal to h, thus d_1 and d_2 cannot both lie in the upper hemisphere or the lower hemisphere. Without loss of generality, let $d_1 \in NH[north]$ and $d_2 \in NH[south]$.

Observation 7.2.3 *If a facet f of P intersects h^+, and f has its outward normal represented by a point q on $\mathcal{S}$, then $q \in NH[d_1]$. Similarly, if f intersects h^-, then $q \in NH[d_2]$. Therefore, if f intersects the casting plane h, then $q \in NH[d_1] \cap NH[d_2]$ (recall that f is open).*

Define $C(d_1)$ and $C(d_2)$ to be the great circles that bound $NH(d_1)$ and $NH(d_2)$. If $\vec{d_1}$ and $\vec{d_2}$ are opposite, then $C(d_1) = C(d_2)$, otherwise, $C(d_1) \cap C(d_2)$ consists of a pair of antipodal points on $\mathcal{S}$ different from *north*, and *south*.

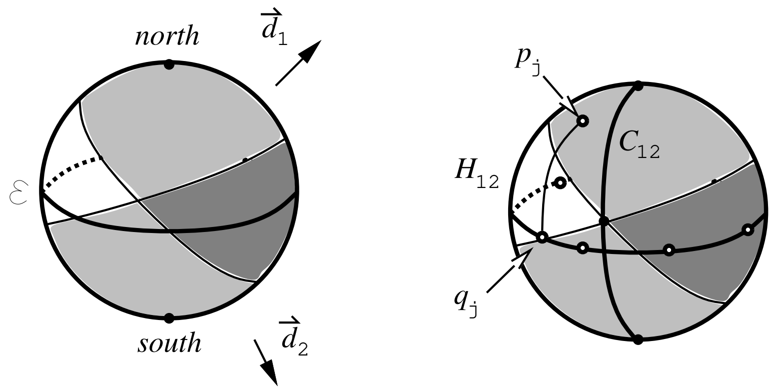

Figure 7.3: The sphere of directions. The shaded hemispheres are $NH(d_1)$ and $NH(d_2)$, and the darker shaded region is their intersection.

For any point $p \in \mathcal{S} - \{north, south\}$, define $\lambda(p)$ to be the nearest point on the equator (i.e., the intersection point of the equator $\mathcal{E}$ with the great circle through *north* and p nearest to p). By definition, we have

$$\overrightarrow{op} \cdot \overrightarrow{o\lambda(p)} \geq 0\,. \tag{7.1}$$

Furthermore, p and $\lambda(p)$ lie to the same side of any great circle through *north* and *south*.

Assume that $\vec{d}_1$ and $\vec{d}_2$ are non-opposite in the following (see Figure 7.3). Define C_{12} to be the great circle containing *north*, *south* and the points of $C(d_1) \cap C(d_2)$. Note that $NH[d_1] \cap NH[d_2]$ does not intersect one of the (open) hemispheres defined by C_{12}. Let H_{12} be this open hemisphere. By the above observation, any facet that has its outward normal in H_{12} cannot be intersected by the casting plane. We use this fact in the following lemma.

Lemma 7.2.1 *If a simple polyhedron P is castable in non-opposite directions with casting plane h, then h contains an edge of P.*

Proof: Let $Q = P \cap h$. If Q consists of more than one connected component, or if Q has holes, then h cannot be a casting plane for P. Therefore, Q is a simple polygon. Let $e_1, \ldots, e_m$ be the clockwise sequence of edges bounding Q and let $q_1, \ldots, q_m$ be the points on $h \cap \mathcal{S}$ that represent the outward normals of $e_1, \ldots, e_m$. Since h is chosen to be horizontal, $q_1, \ldots, q_m \in \mathcal{E}$. Every open half-circle in $\mathcal{E}$ contains at least one point of $q_1, \ldots, q_m$, because Q is a simple polygon.

Given that P is castable with respect to non-opposite directions $\vec{d}_1$ and $\vec{d}_2$, assume that every e_i is the intersection of a facet f_i of P with the casting plane (i.e. no edge of Q is an edge of P). Let C_{12} and H_{12} be as defined above, and let e_j be an edge of Q such that $q_j \in \mathcal{E} \cap H_{12}$ (see Figure 7.3). Let p_j be the point on $\mathcal{S}$ that represents the outward normal of f_j. Then $q_j = \lambda(p_j)$, and by (7.1), we know p_j lies in H_{12}. However, H_{12} does not contain any point in $NH[d_1] \cap NH[d_2]$, so by Observation 7.2.3 the facet f_j cannot intersect the casting plane, which is a contradiction. Thus h contains an edge of P. ∎

Lemma 7.2.2 *If a simple polyhedron P is castable with casting plane h and in non-opposite directions, then h contains an edge of the convex hull of P.*

Proof: Let P be castable with respect to non-opposite directions $\vec{d_1}$ and $\vec{d_2}$. If the cast of $P \cap cl(h^+)$ can be removed in a direction $\vec{d_1}$, then the convex hull of $P \cap cl(h^+)$ can also be removed in the direction $\vec{d_1}$. The same statement holds for direction $\vec{d_2}$ and the cast of $P \cap cl(h^-)$.

Let $Q = P \cap h$. The convex hull of Q is the closure of a facet bounding both $CH(P \cap h^+)$ and $CH(P \cap h^-)$ (note that the convex hull is defined as a *closed* set). As in the proof of the previous lemma, there exists an edge e_j of the convex hull of Q where the outward normal of the edge on plane h lies in H_{12}. We need to prove that e_j is also an edge of $CH(P)$. Let f_1 be the facet of $CH(P \cap h^+)$ incident to e_j and not in h. Define f_2 analogously for $CH(P \cap h^-)$. Let q_j, p_1 and p_2 be the points on $h \cap \mathcal{S}$ and $\mathcal{S}$ that represent the outward normals of e_j, f_1 and f_2, respectively. Since f_1 and f_2 are incident to e_j, we have $\lambda(p_1) = \lambda(p_2) = q_j$, so p_1, p_2 and q_j lie on a half-circle between *north* and *south* and in H_{12}. Since $p_1 \in NH(d_1)$ and $p_2 \in NH(d_2)$ are both contained in H_{12}, the half-circle through *north*, *south*, p_1 and p_2 must contain a point r that is not in $NH[d_1]$ nor in $NH[d_2]$. The plane h' with normal $\overrightarrow{or}$ and containing e_j has $CH(P \cap h^+)$ completely to the one side, with the exception of $cl(e_j)$. Similarly, $CH(P \cap h^-)$ lies completely to the one side of h' with the exception of $cl(e_j)$. Since these convex hulls lie to the same side, it follows that P lies completely to the one side of h' with the exception of the endpoints of e_j, and possibly e_j itself (if e_j is an edge of P). Therefore, e_j is an edge of $CH(P)$. ∎

Notice that the above two lemmas imply that if a polyhedron is castable, but not with opposite cast removal, then the casting plane contains both an edge of P and an edge of the convex hull of P (this might be the same edge). This will aid considerably to determine castability with arbitrary cast removal.

7.2.2 Relation to linear programming

Let P be a polyhedron and let h be a plane. The plane h partitions the set V of vertices of P into three subsets V_h, V_h^+ and V_h^- of vertices in, above and below h, respectively. Similarly, h partitions the set E of edges of P in four subsets E_h, $E_h^\times$, E_h^+ and E_h^- of edges contained in h, intersecting h, above h and below h, respectively. The set F of facets is partitioned in the same way. For any facet $f \in F$, denote by $\Psi(f)$ the closed half-space bounded by a plane supporting f, and such that for any point in f, $\Psi(f)$ does not intersect the interior of P in an ϵ-neighborhood of the point. Denote by $\Psi_0(f)$ the same half-space, but translated such that the bounding plane contains the origin. We define

$$\xi^+(h) \equiv cl(h_0^+) \cap \left\{ \bigcap_{f \in F_h^+ \cup F_h^\times} \Psi_0(f) \right\} \qquad \text{and} \qquad \xi^-(h) \equiv cl(h_0^-) \cap \left\{ \bigcap_{f \in F_h^- \cup F_h^\times} \Psi_0(f) \right\}.$$

The intersection of a set of half-spaces is called *non-trivial* if it contains more than a single point. Denote by *refl*(b) the reflection of an object b through the origin (i.e. every point in b is negated). We make the following observations.

Observation 7.2.4 *The plane h is a casting plane for polyhedron P for arbitrary cast removal if and only if $\xi^+(h)$ and $\xi^-(h)$ are both non-trivial.*

Observation 7.2.5 *The plane h is a casting plane for polyhedron P for opposite cast removal if and only if $\xi^+(h) \cap$ refl$(\xi^-(h))$ is non-trivial.*

Observation 7.2.6 *Let h be a plane and let ℓ be a line perpendicular to h and through the origin. The plane h is a casting plane for polyhedron P for orthogonal cast removal if and only if $\ell \cap \xi^{+}(h) \cap$ refl$(\xi^{-}(h))$ is non-trivial.*

With the above observations, we can test efficiently whether a given plane h is a casting plane for P. Since the casting problem for a plane h and a polyhedron P can be transformed in linear time to a linear programming problem in 3 dimensions, the test requires only linear time [57].

Lemma 7.2.3 *Given a polyhedron P and a plane h, one can test in linear time whether h is a casting plane for P in any of the three versions for removing the cast.*

Similarly, given a polyhedron and two cast removal directions (but not a casting plane), one can test using linear programming whether the polyhedron is castable with respect to those cast removal directions.

Lemma 7.2.4 *Given a polyhedron P and two cast removal directions, one can test in linear time whether there exists a casting plane h that allows removing the cast parts in the given directions.*

Proof: Let the two cast removal directions be $\vec{d_1}$ for $\partial P \cap h^{+}$ and $\vec{d_2}$ for $\partial P \cap h^{-}$. For every facet f of P, one can determine whether f should lie above the casting plane h (is compatible only with $\vec{d_1}$), below h (is compatible only with $\vec{d_2}$), may intersect h (is compatible with both $\vec{d_1}$ and $\vec{d_2}$) or is incompatible with the cast removal directions. If there is a facet of P that is incompatible, then there does not exist any casting plane for directions $\vec{d_1}$ and $\vec{d_2}$.

The classification of the facets as "above", "below", and "intersect" imposes a classification of the edges. Any edge is classified either as "above/above" (a/a), "above/below" (a/b), "above/intersect" (a/i), "below/below" (b/b), "below/intersect" (b/i) or "intersect/intersect" (i/i), corresponding to the classification of the two facets incident to that edge.

Similarly, the classification of an edge determines where both endpoints of the edge must lie. For example, if an edge is classified as (a/a) then both endpoints must lie in $h^{+} \cup h$. We summarize the implications that the classification of the edges has on their endpoints in the table below.

edge class.	endpoints
(a/a)	$h^{+} \cup h$
(a/i)	$h^{+} \cup h$
(b/b)	$h^{-} \cup h$
(b/i)	$h^{-} \cup h$
(a/b)	h
(i/i)	anywhere

The classification of the endpoints of edges, in turn, determines where the vertices of P must lie. Since every vertex is adjacent to at least 3 edges, no vertex can be adjacent to only (i/i) edges. Hence, one can decide for every vertex whether it must be contained in h, lie in $h^{+} \cup h$, or lie in $h^{-} \cup h$. We dualize the vertices to planes, consider the half-spaces to the appropriate side of these planes, based on the classification, and obtain a linear programming problem to decide whether a plane h exists that has the appropriate location with respect to the vertices of P. ∎

7.2.3 Antipodality properties

For opposite cast removal, we prove that if a casting plane intersects a facet, then it intersects the boundary of that facet in antipodal pairs (note that this also holds for orthogonal cast removal). This is an important property that is used to bound the number of distinct casting planes.

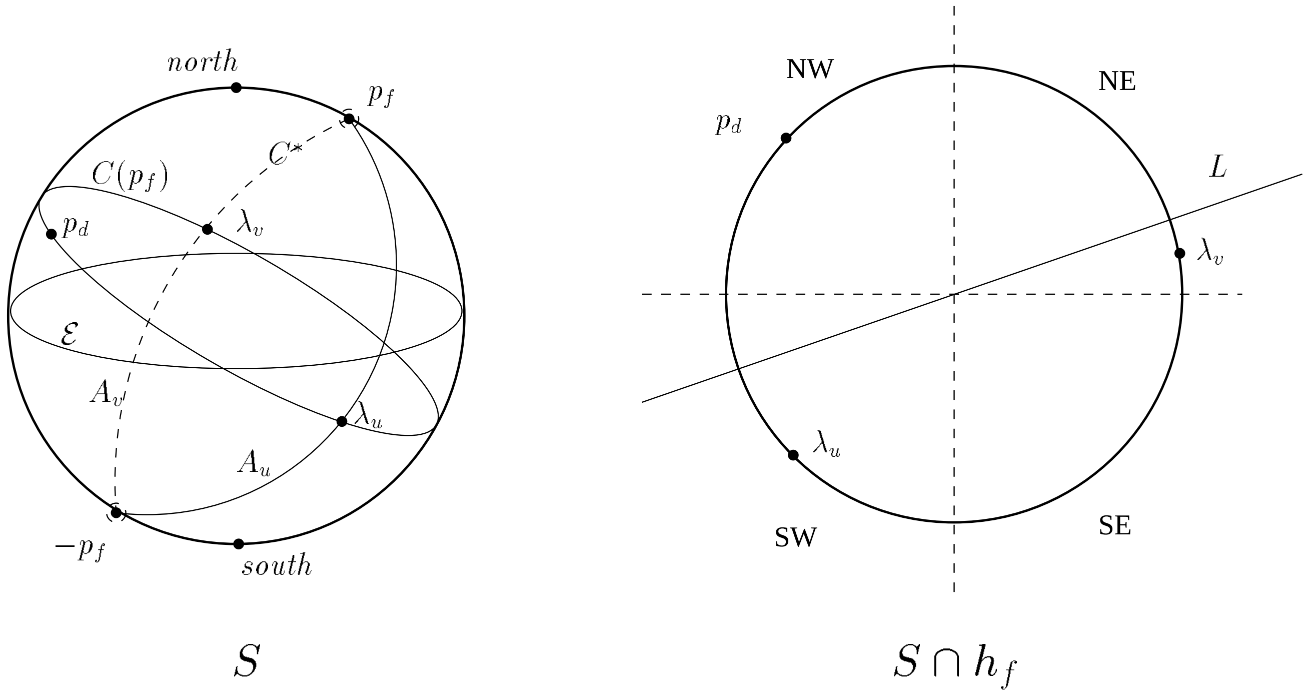

Figure 7.4: Illustrating the proof of Lemma 7.2.5.

Lemma 7.2.5 *If the casting plane h intersects a facet f of a convex polyhedron P, and also two vertices u and v in the closure of f, then for opposite cast removal, vertices u and v must be antipodal in $cl(f)$.*

Proof: Let u, v be two vertices in $cl(f) \cap h$, and assume that they are not antipodal. Let h_f be the plane that contains f. Since u and v are not antipodal, there are two edges e_u and e_v in $cl(f)$ incident to u and v, respectively, which lie on the same side of h and diverge in the plane h_f (when directed away from h). Suppose without loss of generality that $e_u, e_v \in h^+$. Let f_u, f_v be the facets incident to e_u, e_v and different from f.

We again represent the space of all possible directions in 3-space as a sphere of directions with the casting plane as horizontal and *north* $\in h^+$. Let p_f be the point in the (closed) northern hemisphere representing the outward normal of facet f. The inward normal of f corresponds to a point $-p_f$ antipodal to p_f (see Figure 7.4). Since f_u and f_v are each incident to an edge of $cl(f)$, we know that the points representing their facet normals must be on open semi-circles A_u and A_v between p_f and $-p_f$. Let λ_u (respectively λ_v) be the intersection of A_u (respectively A_v) with $C(p_f)$. Let $\vec{d}$ be the casting direction for $P \cap h^+$. By Observation 7.2.3, we know that $\vec{d}$ must correspond to a point p_d on $C(p_f) \cap h^+$.

Consider the great circle $C(p_f) = S \cap h_f$ as a unit circle of directions. Call the semicircle of $C(p_f)$ intersecting h^+ the northern semicircle of $C(p_f)$. Define the southern, eastern, and western semicircles analogously. Since f is a convex polygon, we know that one of λ_u and λ_v must be in the eastern semicircle and one must be in the western semicircle. Without loss of generality, assume that λ_u is in the western semicircle. Since e_u and e_v diverge in h_f, λ_u and λ_v must be strictly south of some non-vertical line L through the origin. This, along with the fact that λ_u and λ_v are split between the eastern and western semicircles, implies that at least one of λ_u and λ_v is in the southern semicircle. Without loss of generality, suppose it is λ_u. Since $\vec{d}$ is by assumption compatible with f_u, p_d must be in the northwest quadrant. Similarly, λ_v must be in the northeast quadrant. Since λ_u and λ_v are strictly south of L, and in opposite quadrants, p_d is incompatible with one of λ_u and λ_v. Without loss of generality suppose p_d is incompatible with λ_v. Recall that $NH(p_d)$ denotes the hemisphere of directions compatible with d. Since a point on A_v (namely λ_v) is outside $NH(p_d)$, and A_v is an arc with its endpoints on $C(p_d)$, all of A_v must be outside $NH(p_d)$. This implies that f_v is incompatible with $\vec{d}$, a contradiction. ∎

Corollary 7.2.1 *Let h be a casting plane for a polyhedron P which intersects a facet f properly, and assume opposite cast removal. If h intersects a vertex v and properly intersects an edge e in the closure of f, then v is antipodal to both endpoints of e. If h properly intersects two edges in the closure of f, then they are parallel.*

7.2.4 Convexity properties

In this subsection we derive some additional geometric properties of convex polyhedra that form the basis of faster algorithms. We also establish an important property that relates the castability of a simple polyhedron to that of its convex hull.

If P is a convex polyhedron, then the linear programming problems defined by P and a candidate casting plane h need not consider all facets of F, but only those intersecting h and those adjacent to h. We make this more precise. For the subset E_h of the edges of P contained in h, let $F^+(E_h)$ denote the subset of F^+ of facets that contain at least one edge of E_h in their closure. Define $F^-(E_h)$ analogously. Furthermore, we define

$$\phi^+(h) \equiv cl(h_0^+) \cap \left\{ \bigcap_{f \in F_h^\times \cup F^+(E_h)} \Psi_0(f) \right\} \qquad \text{and} \qquad \phi^-(h) \equiv cl(h_0^-) \cap \left\{ \bigcap_{f \in F_h^\times \cup F^-(E_h)} \Psi_0(f) \right\}.$$

Lemma 7.2.6 *If P is convex, $\xi^+(h) = \phi^+(h)$ and $\xi^-(h) = \phi^-(h)$.*

Proof: We only prove that $\xi^+(h) = \phi^+(h)$; the other proof is similar. Furthermore, that $\xi^+(h) \subseteq \phi^+(h)$ is trivial, so we prove $\phi^+(h) \subseteq \xi^+(h)$.

If $\phi^+(h)$ only contains the origin then so does $\xi^+(h)$. Otherwise, let r be a half-line originating at the origin and inside $\phi^+(h)$. If $r \notin \xi^+(h)$, then there is a facet $f \in F_h^+ \setminus F^+(E_h)$ for which $r \notin \Psi_0(f)$. Let $\overline{\Psi}(f)$ denote the (closed) half-space supporting f distinct from $\Psi(f)$. Since P is convex,

$$f \subset cl(h^+) \cap \left\{ \bigcap_{f \in F_h^\times \cup F^+(E_h)} \overline{\Psi}(f) \right\}.$$

Since $r \in \phi^+(h)$, it follows that the projection of any point in f parallel to r onto h will lie in $h \cap P$. But since $r \notin \Psi_0(f)$, the line segment connecting a point in f with this projection will be (partially) outside P, namely, in the neighborhood of f. This contradicts the convexity of P. ∎

With Lemma 7.2.6, we conclude the following:

Lemma 7.2.7 *The plane h is a casting plane for a convex polyhedron P for opposite cast removal if and only if $\phi^+(h) \cap \mathit{refl}(\phi^-(h))$ is non-trivial.*

Lemma 7.2.8 *Let h be a plane and let ℓ be a line perpendicular to h through the origin. The plane h is a casting plane for a convex polyhedron P for orthogonal cast removal if and only if $\ell \cap \phi^+(h) \cap \mathit{refl}(\phi^-(h))$ is non-trivial.*

The following theorem forms the crucial link between simple polyhedra and convex polyhedra in terms of castability.

Theorem 7.2.1 *If a simple polyhedron P is castable, then the convex hull of P is also castable using the same casting plane and cast removal directions.*

To prove the theorem, we first establish a few important lemmas.

Lemma 7.2.9 *A convex polyhedron P is a terrain with respect to a facet Q and a direction $\vec{d}$ if and only if the vertices of P project into $cl(Q)$ when projected in direction $-\vec{d}$ onto the supporting plane of Q.*

Proof:
($\Rightarrow$) If P is a terrain with respect to a direction $\vec{d}$ and a facet Q, then every point of P projects into $cl(Q)$ in direction $-\vec{d}$.

($\Leftarrow$) Suppose every vertex of P projects into $cl(Q)$ in direction $-\vec{d}$. Since P is convex, the line segment from every vertex v to Q in direction $-\vec{d}$ must be inside P. It follows that a ray with direction $\vec{d}$ from every vertex is outside P. By Observation 7.2.1, P is a terrain with respect to $\vec{d}$ and Q. ∎

Lemma 7.2.10 *If a polyhedron P is a terrain with respect to a direction $\vec{d}$ and facet Q then $CH(P)$ is a terrain with respect to $\vec{d}$ and $CH(Q)$.*

Proof: Every vertex of P is on one side of the plane induced by Q; it follows that the convex hull of Q must be a facet of $CH(P)$. Since every vertex of $CH(P)$ is a vertex of P, every vertex of $CH(P)$ must project into $CH(Q)$ in direction $\vec{d}$. By Lemma 7.2.9, P is a terrain with respect $\vec{d}$ and $CH(Q)$. ∎

Lemma 7.2.11 *Let h be plane, let C_1 and C_2 be convex polygons in h such that $C_1 \subseteq C_2$, and let S be a set of points entirely contained in one of the half-spaces bounded by h. If $CH(C_1 \cup S)$ is a terrain with respect to a direction $\vec{d}$ and facet C_1, then $CH(C_2 \cup S)$ is a terrain with respect to $\vec{d}$ and C_2.*

Proof: Suppose that $CH(C_1 \cup S)$ is a terrain with respect to $\vec{d}$ and C_1. By Lemma 7.2.9, S projects inside C_1 in direction $-\vec{d}$. Since $C_1 \subseteq C_2$, S also projects inside C_2 in direction $-\vec{d}$. By Lemma 7.2.9, $CH(C_2 \cup S)$ is a terrain with respect to direction $\vec{d}$ and facet C_2. ∎

Proof: (of Theorem 7.2.1)

Let P be a simple polyhedron, and let h be a casting plane for P with casting directions $\vec{d_1}$ for $\partial P \cap cl(h^+)$ and $\vec{d_2}$ for $\partial P \cap cl(h^-)$. The polyhedron $CH(P \cap h^+) \cup CH(P \cap h^-)$ is also castable for casting plane h and directions $\vec{d_1}$ and $\vec{d_2}$ by Lemma 7.2.10. Denote $P^+ = CH(P \cap h^+)$ and $P^- = CH(P \cap h^-)$.

We need to show that $P_H = CH(P)$ is castable with casting plane h and casting directions $\vec{d_1}$ and $\vec{d_2}$. Let $P_H^+ = CH(P_H \cap h^+)$ and $P_H^- = CH(P_H \cap h^-)$. Since P^+ is contained in P_H^+ and P^- is contained in P_H^-, the theorem follows from Lemma 7.2.11. ∎

7.3 The number of distinct casting planes

Given a polyhedron P with vertex set V, two planes h_1 and h_2 are *(combinatorially) distinct* if the partitioning of the facets into F^+, F^-, $F^{\subset}$ and $F^{\times}$ they define is different. By Observation 7.2.2, a trivial upper bound on the number of distinct casting planes for a polyhedron with n vertices is $O(n^3)$.

This section gives a linear upper bound on the maximum number of distinct casting planes for convex polyhedra in case of orthogonal and opposite cast removal as well as a quadratic upper bound for arbitrary cast removal. The proofs are constructive, i.e., sets of *candidate casting planes* of linear or quadratic size are defined which contain all distinct casting planes. In the following sections we will use these sets of candidate casting planes to determine castability efficiently.

7.3.1 Orthogonal and opposite cast removal

Observe that for orthogonal cast removal, a casting plane h can intersect a polyhedron P as follows (these properties follow from the previous section):

1. A facet f that intersects h properly is perpendicular to h.
2. An edge that intersects h properly is perpendicular to h (because otherwise one of the incident facets cannot be perpendicular).
3. Two vertices in the closure of a facet f and in h are antipodal in $cl(f)$. Any vertex and edge in the closure of f and intersecting h are antipodal in $cl(f)$. (See Lemma 7.2.5).

For opposite cast removal, we have the following properties of intersections of a casting plane h and a polyhedron P:

1. The facets of $F^{\times}$ that intersect h properly have their outward normals such that when translated to the origin, they span a plane or part of it (since $\bigcap\{\Psi_0(f) \mid f \in F^{\times}\}$ must contain a line through o).
2. All edges that intersect h properly are parallel (otherwise the incident facets span more than a plane).
3. Any two vertices in the closure of a facet f and in h are antipodal in $cl(f)$. Any vertex and edge in the closure of f and intersecting h are antipodal in $cl(f)$. (See Lemma 7.2.5.)

Let P be a convex polyhedron with n vertices. Since a linear upper bound on the number of distinct casting planes in case of opposite cast removal implies the same result for orthogonal cast removal, we only prove the opposite case.

Lemma 7.3.1 *Given a convex polyhedron P, the number of distinct casting planes that intersect some edge of P properly is at most linear in the number of vertices of P for opposite cast removal.*

Proof: Let E' be a maximal subset of parallel edges of P, and of which at least one edge is properly intersected by some casting plane. By convexity of P, such a casting plane must intersect the closure of all edges of E', because no such closure of an edge can be strictly above or below the casting plane. The cast removal directions are parallel to the edges of E', and by the classification defined in the proof of Lemma 7.2.4, for every vertex v of P it is specified that either $v \in h \cup h^+$ or $v \in h \cup h^-$ or $v \in h$, for any casting plane h. Let V^+, V^- and $V^\subset$ be these three subsets of vertices, respectively. If $V^\subset$ contains three or more vertices, then at most one distinct casting plane is possible for this direction. Otherwise, we consider the following three cases. Note that since P is convex, by Lemma 7.2.6 we only need to consider the facets that intersect h and those adjacent to an edge of P in h.

Case 1: $V^\subset$ is empty. In this case, the facets that intersect h are all the facets adjacent to the edges of E'. Let G^+ be the endpoints of E' contained in V^+ and let G^- be the endpoints of E' contained in V^-. For a plane to intersect the closure of all edges of E', it must separate G^+ from G^-. Since we are considering opposite cast removal, a casting plane must contain at least three vertices. The vertices that the casting plane may contain must come from the set $G = G^+ \cup G^-$, since $V^\subset$ is empty. Therefore, to bound the number of distinct casting planes that intersect an edge of E' properly, we must count the number of planes that separate G^+ from G^- and contain at least three vertices from the set G.

To do this, we dualize the set of vertices G^+ to a set of planes $\mathcal{D}^+$ and the set of vertices G^- to a set of planes $\mathcal{D}^+$. Let I be the convex polytope that lies below all planes in $\mathcal{D}^+$ and above all planes in $\mathcal{D}^-$. The vertices of I are precisely the duals of the planes. Therefore, there are $O(|E_i|)$ distinct planes.

Case 2: $V^\subset$ contains one vertex. Argument similar to case 1. Simply include the vertex in $V^\subset$ in the sets G^+ and G^-.

Case 3: $V^\subset$ contains two vertices. Same argument as case 2.

Thus, we see that the number of distinct casting planes that intersect an edge of E' properly is bounded by $O(|E'|)$. Since every edge of P contributes to only one subset E' of parallel edges, the lemma follows by Euler's formula. ■

The following lemma is the basis of an inductive argument to prove a linear bound on the number of distinct casting planes that intersect no edge properly.

Lemma 7.3.2 *Given a convex polyhedron P, there exists a vertex v with constant degree such that v participates in a constant number of antipodal pairs on the incident facets.*

Proof: Let $\tilde{V}, \tilde{E}, \tilde{F}$ be the number of vertices, edges and facets of P. The summed degree of all vertices $D = 2\tilde{E} \leq 6\tilde{V} - 12$. Every vertex has at least degree 3, thus there must be at least $\tilde{V}/2 + 1$ vertices of degree at most 8. The total number of antipodal pairs, summed over all facets, is at most $3\tilde{F}/2 \leq 3\tilde{V} - 6$, which implies that the total vertex contribution in antipodal pairs, A, satisfies $A \leq 6\tilde{V} - 12$ [69]. Observe that every vertex of P participates in at least 3 antipodal pairs; at least one in each incident facet. If all $\tilde{V}/2 + 1$ vertices of degree at most 8 are in at least 9 antipodal pairs on the incident facets, then $A \geq 9(\tilde{V}/2+1)+3(\tilde{V}/2-1) = 6\tilde{V}+6$, a contradiction. Hence, there exists a vertex which is in at most 8 antipodal pairs and with degree at most 8. ■

Let h be a candidate casting plane of P, and let $Q = h \cap P$. If Q contains three consecutive vertices u, v, w that are also vertices of P, then each of u and w is either an endpoint of an edge incident to v, or a vertex antipodal to v on the closure of a facet f incident to v. We say that the plane through u, v, w is *generated by* v. It follows that the set of candidate casting planes generated by v has size $\binom{d+a}{2}$, where d is the degree of v and a is the number of vertices antipodal to v in the closures of the facets incident to v. Every casting plane h that does not intersect any edge properly contains at least three vertices that are consecutive in $h \cap P$, and therefore, every such casting plane is generated by some vertex of P.

Theorem 7.3.1 *Given a convex polyhedron P with n vertices, the maximum number of distinct casting planes for P is $O(n)$, assuming opposite removal of the cast parts.*

Proof: First, assume that the casting plane h intersects some edge e of P properly. By Lemma 7.3.1, there are $O(n)$ distinct casting planes of this type.

Next, we show that the number of casting planes that do not intersect any edge properly is linear. For such a casting plane h, all vertices of the intersection polygon $Q = h \cap P$ are also vertices of P.

The proof is by induction. Let v be a vertex of P of degree at most 8 and which participates in at most 8 antipodal pairs (see Lemma 7.3.2). The number of casting planes containing v which do not intersect any edges properly is bounded from above by the number of planes generated by v, and hence, is constant. We remove v from P and continue the count on the convex hull of the remaining vertices. We have counted all distinct casting planes that contain v. Since any casting plane of P that does not contain v and does not intersect any edge incident to v properly is also a casting plane of CH(vertices of $P - v$), the lemma follows by induction. ∎

There is another interesting combinatorial bound on the complexity of the intersection of all distinct casting planes with a convex polyhedron. Referring to the proof of Lemma 7.3.1, we notice that two distinct casting planes h_1 and h_2 that intersect an edge of E' properly are *similar*, because they define the same cast removal directions, and they intersect the same closure of edges and facets. In other words, if h_1 and h_2 each intersect edges properly that are parallel, there cannot be two vertices u, v strictly to the one side of h_1 and strictly to different sides of h_2. We use the term *weakly equivalent* for two such planes. Two planes are *strongly distinct* if they are not weakly equivalent. There are $O(n)$ strongly distinct casting planes for any convex polyhedron P with n vertices. We analyze the combinatorial complexity of $h \cap P$, summed over all strongly distinct casting planes h. This quantity is well-defined for opposite cast removal, since two weakly equivalent casting planes have an equal-size intersection with P (although they may intersect different facets, edges and vertices). We prove a bound of $O(n \log n)$ on the summed complexity. Note that when the sum is over all distinct casting planes (not *strongly* distinct), the summed complexity can be $\Theta(n^2)$ if P has a set of $\Omega(n)$ parallel edges. The bound makes use of a hierarchical decomposition of P that closely resembles the hierarchy of Dobkin and Kirkpatrick [26]. It is the basis of the $O(n \log^2 n)$ time algorithms for casting of convex polyhedra with opposite cast removal.

Lemma 7.3.3 *Given a convex polyhedron P with n vertices, there exists a subset V' of the vertices V of size $\Omega(n)$, such that each $v \in V'$ has degree at most 8 and is antipodal to at most 12 vertices in facets incident to v.*

Proof: Similar to Lemma 7.3.2, one can prove that there are at least $\tilde{V}/5$ vertices of degree at most 8 and in at most 12 antipodal pairs. (Otherwise, $A \geq 13(\frac{1}{2} - \frac{1}{5})\tilde{V} + 3(\frac{1}{2} + \frac{1}{5})\tilde{V} = 6\tilde{V}$, a contradiction.) ∎

The following hierarchical decomposition of P generates a set of planes that contains all the candidate casting planes that do not intersect an edge properly. The correctness follows from the proof of Theorem 7.3.1.

Algorithm 11: *Compute all generated planes*

1. Set $i = 1$.
2. Compute the antipodal pairs of the facets of P.
3. Select a subset V_i of V as in Lemma 7.3.3. For every vertex $v \in V_i$, generate all planes through u, v, w. For every vertex $v \in V_i$, the number of generated planes is at most $\binom{12+8}{2} = 190$, thus $O(n)$ for the whole subset.
4. Recompute the convex hull of the vertices of P minus the vertices of V_i.
5. Repeat at step 2 with $i = i + 1$ unless P has no vertices left.

The number of generated planes is linear since each vertex generates a constant number. Antipodal pairs computations take $O(n)$ time and convex hull computations take $O(n \log n)$ time, see e.g. [28, 69]. The total time taken by Algorithm 11 is given by the recurrence $T(n) \leq T((1-\alpha)n) + O(n \log n)$ where $\alpha \geq 1/5$ is the constant in the $\Omega(n)$ of Lemma 7.3.3. This recurrence solves to $T(n) = O(n \log n)$.

Theorem 7.3.2 *Given a convex polyhedron P with n vertices, the total complexity of $h \cap P$, summed over all strongly distinct casting planes h for P, is $O(n \log n)$ for opposite cast removal.*

Proof: In the following proof, we make a distinction between planes that are *generated*, and other planes that can be casting planes. Planes of the second type intersect some edge properly.

Consider a hierarchical decomposition of the vertices of P into sets $V_1, \ldots, V_m$ as described above. Observe that $m \in O(\log n)$.

Let h be any plane, and let $v_1, \ldots, v_k$ be the sequence of vertices in $h \cap P$. We first show that every consecutive subsequence $v_i, \ldots, v_{i+2m-1}$ of vertices that also are vertices of P (no proper intersections of edges of P with h) contains a vertex that generates h. To this end, observe that v_j generates h if and only if v_j is in a vertex set V_s with lower or equivalent index as its neighbors, thus if $v_{j-1} \in V_r$ and $v_{j+1} \in V_t$, then $r \geq s$ and $t \geq s$. Since there are only m vertex sets, any consecutive sequence of $2m$ vertices contain at least one that that generates the plane h.

Consider the vertices v_i that are proper intersections of h and an edge of P. Any edge e gives rise to at most one strongly distinct casting plane, and therefore, the total number of these vertices in $h \cap P$, summed over all strongly distinct casting planes, is linear.

Summarizing, the sequences of $h \cap P$ summed over all strongly distinct casting planes contain $O(n)$ vertices that generate a casting plane, $O(n)$ vertices that are proper intersections of edges with a casting plane, and at most $2m - 1$ vertices in between. It follows that the total complexity of the intersections is $O(nm) = O(n \log n)$. ∎

Corollary 7.3.1 *Given a convex polyhedron P with n vertices, the number of planes that intersect the interior of P but do not intersect any facets of P is $O(n)$, and the number of edges of P contained in these planes, summed over all planes, is $O(n \log n)$.*

7.3.2 Arbitrary cast removal

We have shown that the number of casting planes that also allow opposite cast removal is linear. For the other casting planes, we know from Lemma 7.2.1 that they contain an edge of P. Since we may also assume that they contain a third vertex, we immediately conclude:

Theorem 7.3.3 *Given a convex polyhedron P with n vertices, the number of distinct casting planes for P is $O(n^2)$, assuming arbitrary removal of the cast parts.*

7.4 Algorithms for orthogonal and opposite cast removal

In this section and the next, algorithms are presented for the computation of casting planes, and hence, determining whether a given polyhedron is castable. This section focuses on orthogonal and opposite cast removal.

7.4.1 A simple algorithm for simple polyhedra

We compute $O(n)$ candidate casting planes as follows. By Theorem 7.2.1, we need only consider the casting planes of the convex hull of P. We first compute the candidate casting planes that intersect some edge properly, and then the ones that are generated. We only consider opposite cast removal; the case of orthogonal cast removal only requires some straightforward changes.

Let $E_1, \ldots, E_k$ be a partitioning of E into maximal sets of parallel edges. For each E_i, let V_i^+ denote the upper endpoints of E_i, V_i^- the lower endpoints of E_i, and $V_i^{\subset}$ the set of vertices that must be contained in the casting plane for the cast removal direction parallel to the edges of E_i. We compute all planes that contain the vertices of $V_i^{\subset}$, separate V_i^+ from V_i^-, and contain at least three vertices of $V_i^{\subset} \cup V_i^+ \cup V_i^-$ by intersecting the corresponding set of half-spaces in dual space, as in Lemma 7.3.1. Each vertex of the resulting polyhedron in dual space corresponds to a plane with the desired properties. This gives $O(|E_i|)$ candidate casting planes. The intersection of $|E_i|$ half-spaces in 3-dimensional space can be computed in $O(|E_i| \log |E_i|)$ time, see e.g. [28, 69]. Summed over all subsets $E_1, \ldots, E_k$, we obtain $O(n)$ candidate casting planes in $O(n \log n)$ time.

Second, we compute the other candidate casting planes in $O(n \log n)$ time by Algorithm 11. We conclude:

Lemma 7.4.1 *Given a polyhedron P with n vertices, one can compute in $O(n \log n)$ time a set Γ of $O(n)$ planes such that any casting plane h that contains at least three vertices of P is contained in Γ, assuming opposite cast removal.*

Theorem 7.4.1 *Given a polyhedron P with n vertices, one can decide in $O(n^2)$ time and linear space whether P is castable when the cast parts must be removed in orthogonal or opposite directions.*

Proof: If P is a convex polyhedron, the theorem follows immediately from Lemmas 7.2.3, 7.2.4 and 7.4.1. If P is a simple polyhedron, we additionally apply Theorem 7.2.1. ■

7.4.2 Walking around convex polyhedra

For convex polyhedra, the above result can be improved as follows. By Lemma 7.2.7 determining whether a plane h is a casting plane for P can be done by only considering the facets intersected by h and the facets incident to the edges that are contained in h (this only holds for convex polyhedra). A linear program on this set of facets tells us whether h is a casting plane. We also know, by Theorem 7.3.2, that the total number of facets that we check, for all $O(n)$ candidate casting planes, is only $O(n \log n)$. This will lead to an $O(n \log^2 n)$ time algorithm for a convex polyhedron P with n vertices. The algorithm is split up in two parts, each of which walks around the polyhedron to find the relevant facets. The first algorithm tests each class of weakly equivalent planes that intersect some edge properly. The second tests all remaining planes that are *generated*, in the terminology of Theorem 7.3.2.

Each edge defines a class of weakly equivalent casting planes. The traversal of $h \cap P$ is performed for a generic (i.e. partially specified) plane h in this class. If any plane in the weak equivalence class is a valid casting plane, the linear program constructed by the traversal will find it. By Corollary 7.2.1 we know that any valid casting plane must intersect a facet in antipodal faces. In the next algorithm we take advantage of the fact that if we know the casting direction, and one of the faces of intersection, there is a unique antipodal edge or vertex *any* valid casting plane with this orientation must intersect. We preprocess the polyhedron for Algorithm 12 as follows:

1. With Algorithm 11, compute a hierarchical decomposition of P into $O(\log n)$ vertex sets $V_1, \ldots, V_m$, as in Theorem 7.3.2. Store with each vertex v all $O(1)$ planes generated by v.
2. For every facet f, store the outward normals of the facets that are incident to an edge in the closure of f in a sorted list.
3. For every vertex, store the outward normals of its incident facets in a sorted list (these are linearly ordered since they are incident to the same vertex).

These steps can be done in $O(n \log n)$ time.

Algorithm 12: *Test weak equivalence classes of planes that intersect an edge properly.*

for every edge $e_1 \in E$

  if e_1 is untreated **then**

    Trace $h \cap P$ for a generic casting plane h that intersects all edges parallel to e_1:

    Let $\vec{d}$ be a direction parallel to e_1, and let f_1 be a facet incident to e_1.

    $q_1 \leftarrow e_1$, $i \leftarrow 1$

    repeat

      if q_i is an edge **then**

        mark q_i as treated.

        Let f_{i+1} be the facet adjacent to q_i distinct from f_i.

      else q_i is a vertex

        If (q_{i-1}, q_i, q_{i+1}) is a generated triple, mark it as treated.

        Check (in constant time) if q_i is coplanar with every other vertex discovered; if not then fail.

Find by binary search the facet or edge f_{i+1} distinct from f_i that splits the facets incident to q_i into those compatible with $\vec{d}$ and those incompatible with $\vec{d}$.

end if

Find by binary search the edge or vertex q_{i+1} of f_{i+1} distinct from q_i that splits the edges of $cl(f_{i+1})$ into those where neighboring facet is compatible with $\vec{d}$ and those where it is incompatible.

$i \leftarrow i+1$

until $q_i = e_1$ or h has failed

if the walk returns to e_1 **then**

Determine by linear programming whether a plane exists that intersects the closure of the edges discovered on the walk, and also the discovered vertices (dualize the endpoints of the edges and the vertices as in the proof of Lemma 7.3.1 to obtain the constraints).

If the LP is feasible, polyhedron P is castable with cast removal directions $\vec{d}$ and $-\vec{d}$, and the plane corresponding to the feasible solution of the LP.

end if

end if

next Edge

We now need to test those candidate casting planes that intersect no edge properly. The key observation for the next algorithm is that any casting plane that intersects no edge properly must be generated. For Algorithm 13, we carry out the additional preprocessing steps:

1. For every vertex v of P, store the edges adjacent to v in clockwise order, so that it is possible to determine by binary search for any query plane h containing v, the facets or edges incident to v that h intersects.

2. For every facet f of P, store the vertices in the closure of f in clockwise order, so that it is possible to determine by binary search for any query plane h which edges or vertices in the boundary of f intersect h.

Each of these preprocessing steps can be carried out in $O(n \log n)$ time, so the total preprocessing time is $O(n \log n)$.

For a given candidate casting plane h, we use v_i to denote the i-th vertex of $Q = h \cap P$ discovered, and F_h to to denote the set of facets that intersect h properly or are incident on an edge of P contained in h. It should be noted that triples marked as treated in Algorithm 12 remain marked at the beginning of Algorithm 13.

Algorithm 13: *Test all candidate planes that do not intersect an edge properly*

for every generated triple (u, v_1, v_2)

if (u, v_1, v_2) has not been treated **then**

Let h be the plane through u, v_1, v_2. Mark (u, v_1, v_2) as treated. $i \leftarrow 2$, $F_h \leftarrow \emptyset$.

while we have not walked all the way around to v_1 or failed.

Determine by binary search the edge or facet q_{i+1} that h intersects clockwise from v_i.

if q_{i+1} is an edge $e = (v_i, v)$ **then**

$v_{i+1} \leftarrow v$

Add both facets adjacent to e to F_h.

else q_{i+1} is a facet

Add $f = q_{i+1}$ to F_h.

Determine by binary search what other vertex or edge q' in the boundary of f intersects h.

If q' is an edge, then **next** Triple, since h was tested with Algorithm 12.

Otherwise, $v_{i+1} \leftarrow q'$

end if

if (v_{i-1}, v_i, v_{i+1}) is generated **then**

If (v_{i-1}, v_i, v_{i+1}) has already been treated, h cannot be a casting plane: **next** Triple

Otherwise, mark (v_{i-1}, v_i, v_{i+1}) as treated.

end if

$i \leftarrow i + 1$

next Step

Construct $\phi^+(h)$ and $\phi^-(h)$ from F_h. Test by linear programming if $\phi^+(h) \cap \mathit{refl}(\phi^-(h))$ is non-trivial. If so, accept h as a casting plane, with the casting directions given by the solution to the LP.

end if

next Triple

Theorem 7.4.2 *Given a convex polyhedron P with n vertices, one can decide in $O(n \log^2 n)$ time and linear space whether P is castable when the cast parts must be removed in orthogonal or opposite directions.*

Proof: The above algorithms attain the claimed time bound. This can be seen as follows. The total preprocessing time is $O(n \log n)$. Let us count the total number of steps walking around the polyhedron in Algorithm 12. Since each edge is intersected properly by at most one walk, we charge the step that intersects an edge properly to that edge. Consider the steps between two proper edge intersections. We charge those before the first generated triple encountered to the previous edge properly intersected, and those after the first generated triple to the most recently encountered generated triple. From the proof of Theorem 7.3.2, we know that there are $O(n)$ triples and that every $2m \in O(\log n)$ consecutive vertices contain at least one generated triple. It follows that $O(n \log n)$ steps are charged to generated triples and edges. Since each walking step takes $O(\log n)$ time, Algorithm 12 takes $O(n \log^2 n)$ time to generate linear programs.

For Algorithm 13, the time bound follows in a similar way; each step that discovers a vertex is charged to the most recently encountered generated triple. Since each edge is also discovered by at most one walk in Algorithm 13, we charge that step to the edge. It follows that the second algorithm also takes $O(n \log n)$ steps and $O(n \log^2 n)$ time to generate linear programs.

By Theorem 7.3.2, the total complexity of all linear programs generated by both algorithms is $O(n \log n)$, hence the total time to test all candidate planes is $O(n \log n)$. ∎

7.5 Algorithms for arbitrary cast removal

In this section we study the most general version of the casting problem: determine whether a simple polyhedron P is castable when the cast parts may be removed in arbitrary directions. Using Lemmas 7.2.1 and 7.2.2 and one more observation on arbitrary cast removal, we obtain a simple $O(n^2 \log n)$ time and linear space algorithm.

Let P be a polyhedron. We first test whether P admits opposite cast removal using the simple $O(n^2)$ time algorithm of Theorem 7.4.1. If so, we are done. Otherwise, if P is convex, then, by Lemma 7.2.1, we only have to consider casting planes that contain some edge of P. If P is non-convex, then, by Lemma 7.2.2, we only have to consider casting planes that contain an edge of the convex hull of P.

Observation 7.5.1 *Let P be a polyhedron and h be a plane that contains an edge e of the convex hull of P. Assume without loss of generality that e is horizontal and that a vertical plane exists which supports e and has $P - cl(e)$ completely to the one side.*

- *If $P \cap cl(h^+)$ is a terrain and $P \cap cl(h^-)$ is not a terrain, then no plane μ containing e for which $P \cap h^- \subset \mu^-$ is a casting plane.*
- *If $P \cap cl(h^+)$ is not a terrain and $P \cap cl(h^-)$ is a terrain, then no plane μ containing e for which $P \cap h^+ \subset \mu^+$ is a casting plane.*
- *If $P \cap cl(h^+)$ and $P \cap cl(h^-)$ are both not a terrain, then no plane containing e is a casting plane.*

The above observation sets up a binary search for a casting plane that contains some edge e of the convex hull of P (see Figure 7.5). First, compute the convex hull of P. For any edge e of the convex hull, rotate P such that e is as in the observation. Consider the $n-2$ vertices that are not endpoints of e, and sort them by the order in which a vertical plane supporting e encounters them if the plane starts rotating about e. (The plane h can rotate in two directions about e. It is not important which direction is chosen, as long as this choice is made consistently.) Assume without loss of generality that the order is $v_1, \ldots, v_{n-2}$. We test whether the plane h supporting e and also containing $v_{n/2-1}$ is a casting plane by determining whether $P \cap cl(h^+)$ is a terrain and $P \cap cl(h^-)$ is a terrain. By the above observation, we can stop considering e if both are not terrains. If both are terrains, we can also stop and h is a casting plane. Otherwise, if only $P \cap cl(h^+)$ is a terrain, we continue the binary search on $v_{n/2}, \ldots, v_{n-2}$, and if only $P \cap cl(h^-)$ is a terrain, we continue the binary search on $v_1, \ldots, v_{n/2-2}$. After at most $\lceil \log_2(n-2) \rceil$ steps, we have determined whether there exists a casting plane that contains e. This leads to:

Theorem 7.5.1 *Given a simple polyhedron P with n vertices, one can determine in $O(n^2 \log n)$ time and linear space whether a casting plane for P exists, when the cast parts can be removed in arbitrary directions.*

Proof: To decide whether opposite cast removal is possible we first apply Theorem 7.4.1 and use $O(n^2)$ time. The computation of the convex hull of P requires $O(n \log n)$ time. There are $O(n)$ edges about which a plane is rotated. The sorting of the vertices $v_1, \ldots, v_{n-2}$ takes $O(n \log n)$ time, and each step of the binary search takes linear time by Lemma 7.2.3. Hence, the above procedure takes $O(n^2 \log n)$ time. ∎

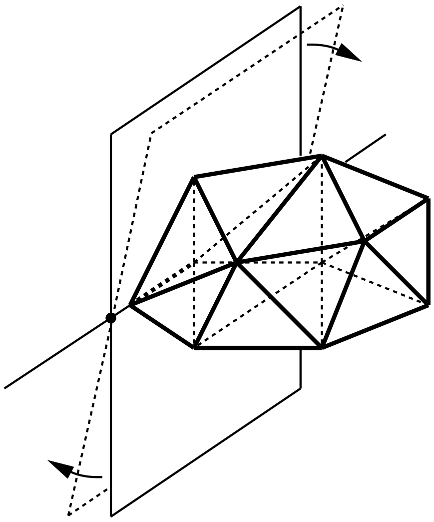

Figure 7.5: Rotating a plane about an edge through P.

7.6 Discussion

In this chapter, we addressed the geometric version of the problem of determining whether a simple polyhedron can be manufactured using casting, and simple algorithms that use $O(n^2)$ or $O(n^2 \log n)$ time and linear space based on linear programming. These algorithms can be improved theoretically using partition trees and their variants[52]. However, we have not presented these improvements since we are mainly concerned with practical algorithms for casting. A detailed discussion on the theoretical improvements can be found in [15]. We summarize our results along with the theoretical improvements in the table below.

		orthogonal	opposite	arbitrary
linear space	convex polyhedra	$O(n \log^2 n)$	$O(n \log^2 n)$	$O(n^2 \log n)$
	simple polyhedra	$O(n^2)$	$O(n^2)$	$O(n^2 \log n)$
best results (in theory)	convex polyhedra	$O(n \log^2 n)$	$O(n \log^2 n)$	$O(n^{3/2+\epsilon})$
	simple polyhedra	$O(n^{3/2+\epsilon})$	$O(n^{3/2+\epsilon})$	$O(n^{3/2+\epsilon})$

We note that more complicated objects can be made by using cores and inserts [31, 71, 87, 88]. Their use slows down the manufacturing process and makes it more costly, and therefore, should be avoided. However, some objects cannot be made without the use of cores and inserts. It would be interesting to develop algorithms that can determine when objects are castable with the use of cores and inserts.

Chapter 8

Conclusions

In this thesis, we have explored the geometric aspects of a few fundamental problems stemming from the manufacturing industry. The problems are taken from two complimentary categories of manufacturing processes: rapid prototyping systems and casting processes. This investigation only scratches the surface of the vast number of applications of geometry in manufacturing. There exist many unanswered questions related to the topics covered in this thesis, and many related areas remain unexplored. We conclude with a list of open problems suggested by our research.

Questions related to Manufacturing Processes

1. In the variable-angle stereolithography model, we assumed that an object cannot be built on a vertex or edge since the object would not be stable. However, in practice, objects may be constructed on a vertex or edge by introducing support stilts as the object is being built in order to maintain stability. It would be interesting to incorporate this into the variable-angle model.

2. What is the maximum number of distinct casting planes in case of arbitrary cast removal? We show an upper bound of $O(n^2)$, whereas the only lower bound we have is linear.

3. For a convex polyhedron P, what is the maximum summed complexity of the intersection of all distinct casting planes with P? We show an upper bound of $O(n \log n)$ in case of opposite cast removal, but the trivial lower bound is linear.

4. Give simple algorithms for casting that improve our simple $O(n \log^2 n)$, $O(n^2)$ and $O(n^2 \log n)$ time algorithms.

5. Suppose that the casts may be removed with any motion. Give algorithms to determine whether a polyhedron is castable in this case.

6. Suppose that we wish to determine castability of an object with non-linear boundaries. Give (simple) algorithms that solve this problem.

7. Suppose that more cast parts are allowed. Determine for a polyhedron how many cast parts are necessary.

8. Related to the previous problem, determine how many cores and inserts are needed to manufacture an object by casting or molding.

9. For some casting processes, it is not necessary that the cast parts be separated by a plane. In these cases, every convex polyhedron is castable. However, no algorithms are known for cast removal of simple polyhedra.

10. We only presented algorithms for computing suitable locations of pin gates in polygonal molds. Generalizing this to polyhedral molds would be interesting. It would also be interesting to find approximate solutions for polyhedral molds, as finding exact solutions seems difficult.

11. Reduce the time complexity of Algorithm 3, which finds the geodesic center of P constrained to lie in Q. The time complexity is $O(n(n+k))$ but it seems like $O(n \log n + k)$ or at least $o(n^2 + k)$ should be possible.

12. The algorithms to compute the optimal orientation of a mold as well as the ones to locate a suitable pin gate have not been tested experimentally.

13. Generalize the algorithms to compute the optimal orientation of a mold to handle more complex objects, such as objects with non-linear boundaries, or of higher genus.

14. There are many other manufacturing processes that have not been analyzed from this perspective such as processes with centrifugal forces for filling, or laser sculpting.

Bibliography

[1] Akin, J.E., *Computer-Assisted Mechanical Design.* Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[2] Anselman, G.W., J. Cunningham, R.A. Green, J.C. Lee, E.H. Phelps, A.M. Prewitt, V. Rowell, L.E. Taylor, C.W. Ward, E.L. Kotzin (editors), *Analysis of Casting Defects*, American Foundrymen's Society, Des Plaines, Ill., 1974.

[3] Aronov. B., S. Fortune, and G. Wilfong, The Furthest-Site Geodesic Voronoi Diagram. *Discrete and Computational Geometry*, **9**, pp. 217-255, 1993.

[4] Asano, T. and G.T. Toussaint, Computing the Geodesic Center of a Simple Polygon, in *Perspectives in Computing: Discrete Algorithms and Complexity, Proceedings of Japan-US Joint Seminar*, D.S. Johnson, A. Nozaki, T. Nishizeki, H. Willis, Eds, Academic Press, Boston, pp. 65-79, 1986.

[5] Asberg, B., G. Blanco, P. Bose, J. Garcia, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu, Feasibility of Design in Stereolithography. in *Proc. 13th Symp. on FST & TCS*, 1993, also available as: Technical Report No. TR-SOCS-94.7, School of Computer Science, McGill University, 1994.

[6] Ashley, S., Rapid Prototyping Systems. *Mechanical Engineering*, **113**(4), pp. 34-43, 1991.

[7] Avis, D. and G. Toussaint, An Optimal Algorithm for Determining the Visibility of a Polygon from an Edge, *IEEE Transactions on Computers,* **C-30-12**, pp. 910-914, 1981.

[8] Berk, A.A., *Computer Aided Design and Analysis for Engineers.* Blackwell Scientific Publications, Oxford, England, 1988.

[9] Besant, C.B., and C.W.K. Lui, *Computer-Aided Design and Manufacture.* Ellis Horwood Limited, West Sussex, England, 1986.

[10] Bhattacharya, B., and G. Toussaint, On Geometric Algorithms that Use the Furthest-Point Voronoi Diagram, in *Computational Geometry*, G. Toussaint, Ed., North Holland, Amsterdam, pp. 43-62, 1985.

[11] Bondy, J. and U.S.R. Murty, *Graph Theory with Applications.* Elsevier Science, New York, New York, 1976.

[12] Bose, P. and G. Toussaint, Geometric and Computational Aspects of Injection Molding. *Proc. Third International Conf. on CAD and Computer Graphics*, August 1993, Beijing, China, pp. 237-242, also available as Technical Report No. SOCS 92.16, School of Computer Science, McGill University, 1992.

[13] Bose, P. and G. Toussaint, Geometric and Computational Aspects of Gravity Casting, to appear in C.A.D.

[14] Bose, P., M. van Kreveld and G. Toussaint, Filling Polyhedral Molds, *Proc. Third WADS* (1993), Lecture Notes in Computer Science 709, Springer-Verlag, pp. 210-221, also available as Technical Report No. SOCS 93.1, School of Computer Science, McGill University, 1993.

[15] Bose, P., D. Bremner, and M. van Kreveld, Castability of Simple Polyhedra, in *Proceedings of the of the 10th ACM Symposium on Computational Geometry*, Stony Brook, NY, pp. 123-131, 1994, also available as Technical Report No. SOCS 93.12, School of Computer Science, McGill University, 1993.

[16] Bose, P., T. Shermer, G. Toussaint, and B. Zhu, Guarding Polyhedral Terrains, *Proc. Allerton Conf.*, Urbana-Champaign, Ill., October 1992.

[17] Bown, J., *Injection Moulding of Plastic Components*, McGraw-Hill, England, 1979.

[18] Chan, T., A Simple Trapezoidal Sweep Algorithm for Reporting Red/Blue Segment Intersections, in *Proc. of the Sixth Canadian Conference on Computational Geometry*, Saskatoon, Saskatchewan, pp. 263-268, 1994.

[19] Chazelle, B., Triangulating a Simple Polygon in Linear Time, *Discrete and Computational Geometry*, **6**, pp. 485-524, 1991.

[20] Chazelle, B., and H. Edelsbrunner, An Optimal Algorithm for Intersecting Line Segments in the Plane. *J. ACM* **39**, pp. 1–54, 1992.

[21] Clarkson, K.L., New Applications of Random Sampling in Computational Geometry. *Discr. & Comp. Geom.* **2**, pp. 195–222, 1987.

[22] Clarkson, K. L., and P. W. Shor, Applications of Random Sampling in Computational Geometry, II. *Discr. & Comp. Geometry* **4**, pp. 387–421, 1989.

[23] Cole, R. and M. Sharir, Visibility Problems for Polyhedral Terrains. *Journal of Symbolic Computation*, **7**, pp.11-30, 1989.

[24] Connolly, R., Manager Injection Molding, Industrial Materials Institute, National Research Council of Canada, Boucherville, Quebec, personal communication, November 1993.

[25] Djidjev, H.N., A. Lingas, and J. Sack, An $O(n \log n)$ Algorithm for Computing the Link Center of a Simple Polygon. *Discrete and Computational Geometry*, **8**, pp. 131-152, 1992.

[26] Dobkin, D. P., and D. G. Kirkpatrick, A Linear Time Algorithm for Determining the Separation of Convex Polyhedra. *J. Algorithms* **6**, pp. 381–392, 1985.

[27] Dobkin, D., and S. Reiss, The Complexity of Linear Programming. *Theoretical Computer Science,* **11**, pp. 1–18, 1980.

[28] Edelsbrunner, H., *Algorithms in Combinatorial Geometry.* Springer-Verlag, Berlin, 1987.

[29] Edelsbrunner, H., and L. Guibas, Topologically Sweeping an Arrangement. *J. Comp. Sys. Sciences* **38**, pp. 165–194, 1989.

[30] ElGindy, H. and D. Avis, A Linear Algorithm for Computing the Visiblity Polygon from a Point, *Journal of Algorithms,* **2**, pp. 186-197, 1981.

[31] Elliott, R., *Cast Iron Technology.* Butterworths, London, 1988.

[32] ElWakil, S.D., *Processes and Design for Manufacturing*, Prentice Hall, New Jersey, 1989.

[33] Elzinga, D.J., and D.W. Hearn, Geometrical Solutions for some Minimax Location Problems, *Transportation Science*, **6**, pp. 379-394, 1972.

[34] Elzinga, D.J., and D.W. Hearn, The Minimum Covering Sphere Problem, *Management Science*, **19**(1), pp. 96-104, 1972.

[35] Fekete, S.P. and J.S.B. Mitchell, Geometric Aspects of Injection Molding, State University of New York, Stony Brook, manuscript 1992.

[36] Flavelle, D., Personal Factories on Computer Horizon, article in *The Gazette*, Montreal, pp. C4, Wednesday, July 28, 1993.

[37] Gajentaan, A., and M.H. Overmars, *$O(n^2)$ Difficult Problems in Computational Geometry.* Tech. Report, Dept. of Computer Science, Utrecht University, 1993.

[38] Ghosh, S.K., Computing the Visibility Polygon from a Convex Set and Related Problems. *Journal of Algorithms*, **12**, pp. 75-95, 1991.

[39] Guibas, L., and J. Hershberger, Optimal Shortest Path Queries in a Simple Polygon. *Journal of Computer and System Sciences*, **39**, pp. 126-152, 1989.

[40] Guibas, L., J. Hershberger, D. Leven, M. Sharir, and R. Tarjan, Linear-Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons. *Algorithmica*, **2**, pp. 209-233, 1987.

[41] Hearn, D.W., J. Vijay, Efficient Algorithms for the (Weighted) Minimum Circle Problem, *Operations Reserach*, **30**, pp. 777-795, 1982

[42] Held, M., *On the Computational Geometry of Pocket Machining.* Lect. Notes in Comp. Science 500, Springer-Verlag, 1991.

[43] Horn, A. and F.A. Valentine, Some Properties of L-Sets in the Plane, *Duke Mathematics Journal*, **16**, pp. 131-140, 1949.

[44] Hudson, P.C, and M.J. O'Carroll, editors, *Mathematical Modelling of Industrial Processes.* Emjoc Press, North Yorkshire, England, 1983.

[45] Kirkpatrick, D., Optimal Search in Planar Subdivisions, *SIAM Journal of Computing*, **12**(1), pp. 28-35, 1983.

[46] van Kreveld, M., *Fat Dissection and Covering, and the Union Size of Polygons*, Technical Report No. SOCS-93.2, School of Computer Science, McGill University, 1993.

[47] Lyman, T. (editor), *Casting Design Handbook*, American Society for Metals, Ohio, 1962.

[48] Lawson, C.L., The Smallest Covering Cone or Sphere, *SIAM Review*, **7**(3), pp. 415-417, 1965.

[49] Lee, C.C. and D.T. Lee, On a Circle-Cover Minimization Problem, *Information Processing Letters*, **18**, pp. 109-115, 1984.

[50] Lee, D.T. and F.P. Preparata, An Optimal Algorithm for Finding the Kernel of a Polygon, *SIAM Journal of Computing*, **6**(3), pp. 594-606, 1979.

[51] Lenhart, W., R. Pollack, J. Sack, R. Seidel, M. Sharir, S. Suri, G. Toussaint, S. Whitesides, and C. Yap, Computing the Link Center of a Simple Polygon. *Discrete and Computational Geometry*, **3**, pp. 281-293, 1988.

[52] Matoušek, J., Efficient Partition Trees. *Discr. & Comp. Geometry* **8**, pp. 315–334, 1992.

[53] Matoušek, J., Range Searching with Efficient Hierarchical Cuttings. *Discr. & Comp. Geometry* **10**, pp. 157–182, 1993.

[54] Matoušek, J., N. Miller, J. Pach, M. Sharir, S. Sifrony, and E. Welzl, Fat Triangles Determine Linearly Many Holes. *Proc. 32nd IEEE Symp. Found. Comp. Science*, pp. 49–58, 1991.

[55] McAllister, M., and J. Snoeyink, Two Dimensional Computation of the Three Dimensional Reachable Region for a Welding Head. *Proc. 5th Canadian Conf. on Comp. Geom*, St. John's, Newfoundland, pp. 437-442, 1993.

[56] McCallum, D. and D. Avis, A Linear Time Algorithm for Finding the Convex hull of a Simple Polygon. *Inform. Process. Lett.* 8, pp. 201–206, 1979.

[57] Megiddo, N., Linear Time Algorithms for Linear Programming in R^3 and Related Problems, *SIAM Journal of Computing*, **12**(4), pp. 759-776, 1983.

[58] Megiddo, N., Linear Programming in Linear Time when the Dimension is Fixed.*J. ACM* **31**, pp. 114–127, 1984.

[59] Melkman, A., On-line Construction of the Convex Hull of a Simple Polyline. *Inform. Process. Lett.* **25**, pp. 11–12, 1987.

[60] Melville, R.C., An Implementation Study of Two Algorithms for the Minimum Spanning Circle Problem, in *Computational Geometry*, G. Toussaint, Ed., North Holland, Amsterdam, pp. 267-294, 1985.

[61] Mulmuley, K., A Fast Planar Partition Algorithm, I. *Proc. 29th IEEE Symp. Found. Comp. Science*, pp. 580-589, 1988.

[62] Okabe, A., B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, England, 1992.

[63] O'Rourke, J., *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, New York, 1987.

[64] O'Rourke, J., *Computational Geometry in C*, Cambridge University Press, New York, 1994.

[65] Pollack, R., M. Sharir, and G. Rote, Computing the Geodesic Center of a Simple Polygon. *Discrete and Computational Geometry*, **4**, pp. 611-626, 1989.

[66] Preparata, F., Minimum Spanning Circle, in *Steps into Computational Geometry*, F.P. Preparata, Ed., University of Illinois, Urbana, Ill., pp. 3-5, 1977.

[67] Preparata, F. and S. Hong, Convex Hulls of Finite Sets of Points in Two and Three dimensions, *Communications of the ACM*, **2,20**, pp. 87-93, 1977.

[68] Preparata, F., and D. Muller, Finding the Intersection of n Half-spaces in Time $O(n \log n)$. *Theoretical Computer Science,* **8**, pp. 45–55, 1979.

[69] Preparata, F.P. and M.I. Shamos, *Computational Geometry – An Introduction.* Springer-Verlag, New York, 1985.

[70] Preparata, F.P. and K.J. Supowit, Testing a Simple Polygon for Monotonicity, *Information Processing Letters*, **12**(4), pp. 161-164, 1981.

[71] Pribble, W.I., Molds for Reaction Injection, Structural Foam and Expandable Styrene Molding, in *Plastics Mold Engineering Handbook, Fourth Edition*, J. Harry DuBois and Wayne I. Pribble (eds.), Van Nostrand Reinhold Company Inc., New York, 1987.

[72] Rademacher, H. and O. Toeplitz, *The Enjoyment of Mathematics*, Princeton University Press, Princeton, N.J., 1957.

[73] Rosenbloom, A. and R. Rappaport, Moldable and Castable Polygons, *Proc. of the Fourth Canadian Conference on Computational Geometry*, St. John's, Newfoundland, pp. 322-327, 1992.

[74] Sack, J.R. and S. Suri, An Optimal Algorithm for Detecting Weak Visibility of a Polygon, *IEEE Transactions on Computers*, **39**, 1990.

[75] Sarnak, N. and R.E. Tarjan, Planar Point Location using Persistent Search Trees, *Communications of the ACM*, **29**(7), pp. 669-679, 1986. .

[76] Shamos, M.I., *Computational Geometry*, Ph.D. Thesis, Yale University, May 1978.

[77] Shamos, M.I., and D. Hoey, Closest-point Problems, in proceedings of the *Sixteenth Annual IEEE Symposium on Foundations of Computer Science*, pp. 151-162, 1975.

[78] Suri, S., On Some Link Distance Problem in a Simple Polygon. *IEEE Transactions on Robotics and Automation*, **6**(1), pp. 108-113, 1990.

[79] Suri, S., Computing Geodesic Furthest Neighbors in Simple Polygons. *Journal of Computer and System Sciences*, **39**, pp. 220-235, 1989.

[80] Seidel, R., Small-dimensional Linear Programming and Convex Hulls Made Easy. *Discr. & Comp. Geometry* **6**, pp. 423–434, 1991.

[81] Simpson, B.L., *History of the Metal Casting Industry*, American Foundrymen's Society Inc., Il., 1969.

[82] Sylvester, J.J., On Poncelet's Approximate Linear Valuation of Surd Forms, *Phil. Mag.*, **4**(20), pp. 203-222, 1860.

[83] Tanner, J.P., *Manufacturing Engineering: An Introduction to the Basic Functions.* Marcel Dekker, New York, New York, 1991.

[84] Toregas, C., R. Swain, C. Revelle, and L. Bergman, The Location of Emergency Service Facilities, *Operations Research*, **19**, pp. 1363-1373, 1971.

[85] Toussaint, G., Movable Separability of Sets, *in Computational Geometry*, North-Holland, New York, pp. 335–375, 1990.

[86] Trochu, F., personal communication, November 1993.

[87] Walton, C.F., and T.J. Opar (Eds.), *Iron Castings Handbook.* Iron casting society, Inc., 1981.

[88] Whelan, A., *Injection Moulding Machines.* Elsevier, London, 1984.