Geometric Aspects of the Casting Process

Geometrische Aspecten van het Casting Proces
(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van
doctor aan de Universiteit Utrecht
op gezag van de Rector Magnificus, Prof. Dr. W. H. Gispen
ingevolge het besluit van het College voor Promoties
in het openbaar te verdedigen
op dinsdag 4 december 2001 des ochtends te 10:30 uur

door

Hee-Kap Ahn
geboren op 3 april 1973, te Daegu (Zuid-Korea)

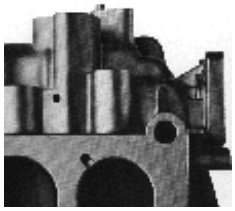
promotor: Prof. Dr. M.H. Overmars
Faculteit Wiskunde en Informatica
co-promotor: Dr. O. Cheong
Faculteit Wiskunde en Informatica

ISBN 90-393-2869-2

Contents

Part I

Introduction



CHAPTER 1

Casting and computer-aided design systems

Manufacturing [31] is the process of converting raw materials (such as iron, glass or polymer) into useful products, ranging from goods such as kettles and telephones to machinery such as railway locomotives and aircrafts. Computer-aided design (CAD) and computer-aided manufacturing (CAM) have automated these manufacturing processes, both in the design phase and the construction phase. Due to the geometric nature of manufacturing processes, many geometric problems arise in the automation of manufacturing. Computational geometry arises at all levels of manufacturing, from design, modeling and simulation to process planning, on-line verification and testing. The survey by Bose and Toussaint [11, 14] gives an overview of geometric problems and algorithms relevant to manufacturing.

In this thesis we study some geometric aspects of the *casting process*, a commonly used manufacturing process for plastic and metal objects, and give algorithms to solve several geometric problems arising in casting. This introduction provides the necessary background, overview, and definitions to appreciate the following chapters of this thesis.

Section 1.1 gives a brief introduction to manufacturing, and introduces several processes in the manufacturing industry. We briefly introduce casting, stereolithography and extrusion.

Section 1.2 introduces the casting process. We introduce sand casting, injection moulding and die casting. The adequate process is chosen depending on factors such as the material, the feeding system for the material, required quality standards, whether the object will be

mass-produced, and so on.

Section 1.3 shows how computers have become an essential element in the manufacturing process, from primitive systems for 2-dimensional drawing and drafting (the *Sketchpad* system of the early 1960s) to the current sophisticated systems for 3-dimensional modeling and simulation.

Section 1.4 introduces geometric aspects of the casting process. The fundamental question arising during the design of an object is whether the object can actually be manufactured using a casting process. We focus on a geometric decision at the basis of the problem and define the problem we address in this thesis. We also briefly introduce features of an object that facilitate the geometric decision process.

Sections 1.5–1.9 provide an overview on the five geometry problems that are studied in Chapters 3–7. We give definitions of problems and summarize our results.

1.1 Manufacturing processes

Manufacturing industries have been considered one of the competitive technologies in today's economy. Even though the word *manufacture* itself comes from the Latin words *manus* and *facere* meaning "to make by hand," most products today are mass-produced with the help of machines.

There are many different production processes for constructing objects in the manufacturing industry, including gravity casting, injection moulding [15, 21, 45], layered manufacturing (as, for instance, stereolithography [9]), material removal via conventional (or chemical or electrical) machining [25], deformation (forging, rolling, extrusion, bending), composition (as in composite materials, sintered ceramics, and the like), and spray deposition.

The casting process is used extensively to mass-produce a wide variety of products. In the process, liquid is fed into a cast (mould) that has a cavity with the shape of the object to be manufactured. After the liquid has hardened, the cast parts are removed from the object. Depending on the material and the feeding system for the molten material (either using gravity or by force), different casting methods are used. More details are discussed in Section 1.2.

Stereolithography [9] is a layer-deposition manufacturing process using a vessel of photo sensitive liquid plastic, a table controlled by a computer, and a laser (see Figure 1.1). The laser lies above the vessel and shines on the surface of the liquid plastic. At the first step the table in the vessel is just below the surface of plastic. The laser moves horizontally, solidifying this layer of plastic. This is the bottom-most layer of the object. At the next step the table is lowered a little bit so that liquid covers the hardened layer, and the laser then draws the next layer. This process is repeated for subsequent layers until the entire

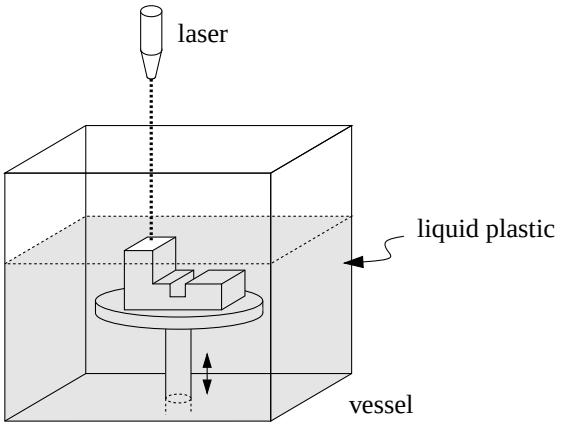


Figure 1.1: Stereolithography

object is formed.

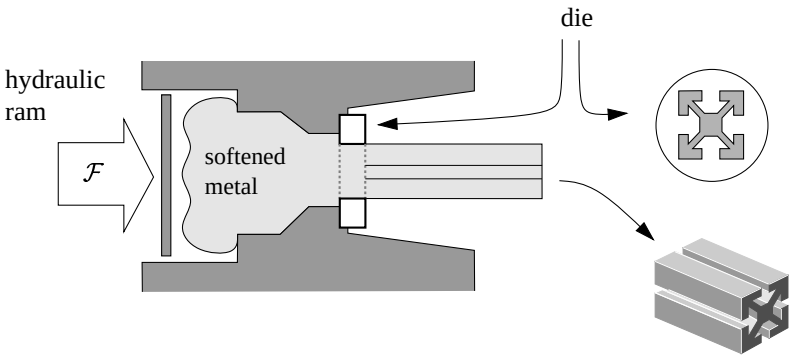


Figure 1.2: Extrusion

The extrusion process is a widely spread manufacturing technology to produce parts used mainly in the construction industry (such as PVC window profiles, pipes and tubes), automotive applications (such as rubber seals and gas conducts), biomedical applications (such as medical tubings), etc. The material is softened by heat prior to extrusion. The heated material is placed into an extrusion press, where a powerful hydraulic ram or a rotating screw forces the softened material through a precision opening, known as a die, to produce the desired shape (See Figure 1.2). Bakers, for example, use a collection of

shaped nozzles to decorate cakes with fancy bands of icing. They are producing extruded shapes. As suggested by these nozzles, the shape of the extrusion is determined by the shape of the opening (die).

1.2 The casting process

The casting process has been widely used for a long time to make household utensils, kitchenware, works of art, etc. For example, the bronze statue of Zeus shown in Figure 1.3 was made using the casting process in 470 BC. Figure 1.4 shows the process in which

Figure 1.3: Zeus throwing lightnings, Bronze, ca. 470 BC

this statue was made by hand: A sculptor carved a prototype of the statue in wax. The prototype was covered by clay, leaving a pin gate. To make a cavity inside the clay mould, the wax prototype was molten and passed away through the pin gate. Molten bronze was poured into the cavity with the shape of Zeus using gravity as in Figure 1.4 (c). After the bronze had hardened the clay mould was broken. As shown in Figure 1.4, the cast itself is broken at the end of the process, and to make another duplicate one had to restart from the beginning, making a new cast.

Today, the prevailing mode of production is called “mass production”: one wants to reuse

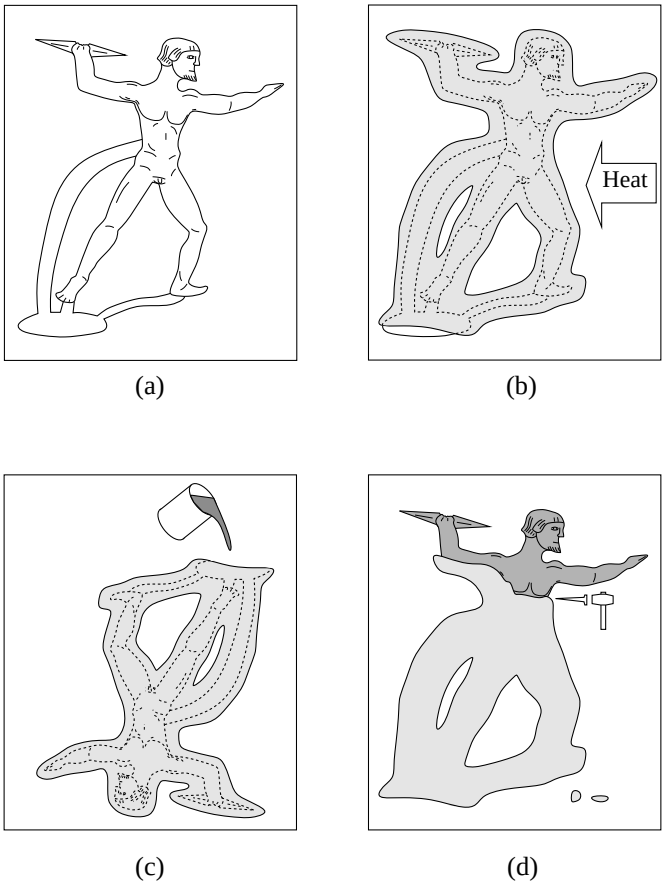


Figure 1.4: The casting process of old days: making a bronze statue of the Zeus: (a) prototype in wax, (b) the prototype is completely covered by clay, after which wax melts and comes out of the clay cover leaving cavity, (c) molten bronze is poured into the cavity, (d) the outer clay cover is broken to get the bronze statue.

the cast many times to produce identical objects.

The industrial casting process consists of two stages. First, liquid is filled into a cavity formed by two cast parts. After the liquid has hardened, one cast part retracts, carrying the object with it. Afterwards, the object is ejected from the retracted cast part (see Figure 1.5). In both retraction and ejection steps, the cast parts and the object should not be damaged, so that the quality of final object is guaranteed and the cast parts can be reused to produce another object.

Depending on the materials (iron, aluminum, polymer, zinc, etc.) being used, the mass

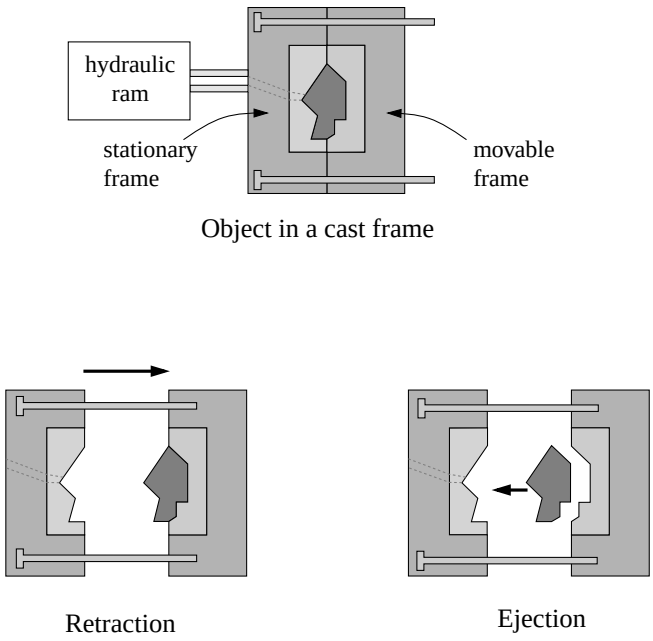


Figure 1.5: The casting process in the real world

producibility of moulds, and the feeding systems (gravity or pressure), there are many different methods for the process.

Most castings of metals, especially large ones, are made in *sand moulds*. In *sand casting* a prototype of the object is first obtained, after which the prototype is divided into two parts along a plane (called *a parting plane*). Sand, mixed with a binder to hold it together, is pressed around the prototype. The sand mould is divided into two along the parting plane, and the prototype is removed from the mould leaving a cavity in the sand mould. Then two sand mould parts are placed together along the parting plane (See Figure 1.6 (c)). To build an object, liquid metal is poured into the cavity through the pin gate using gravity. After the metal has solidified, the sand mould is usually broken and leaves the object.

Injection moulding is a method of casting where plastic is forced into a mould cavity under pressure. The cavity is filled with plastic, and the plastic changes phase to a solid, resulting in an object. Because of the high pressures involved, the mould must be clamped shut during injection and cooling.

Large numbers of small, precise metal parts that have a low melting point, such as zinc, are made by *die casting* using permanent steel moulds. Die casting is accomplished by

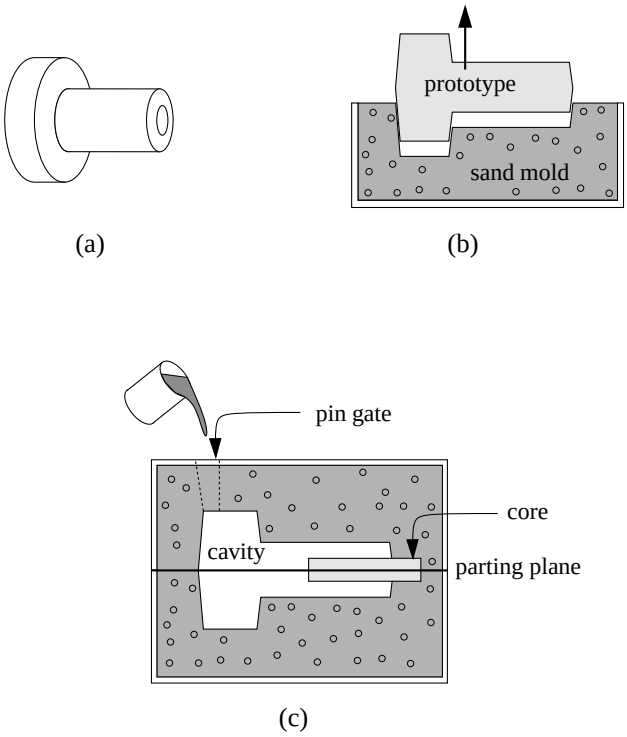


Figure 1.6: The process of sand casting. (a) the object to be cast, (b) prototype leaves a cavity, (c) the cross-section of a typical two-part sand mould

forcing molten metal alloy into a steel mould under high pressure. The heat from the molten metal flows by conduction into the steel mould, which causes the molten metal to solidify. The process is often described as "the shortest distance between raw material and finished product." Unlike sand casting, die casting is used for mass-producing high quality objects, such as handles, brackets, camera bodies and telephone parts, with high speed.

1.3 Automated computer-aided design systems

Computer-aided manufacturing automates manufacturing processes by letting computers communicate instructions directly to the manufacturing machinery. A single computer can control banks of robotic milling machines, lathes, welding machines, and other tools, moving the product from machine to machine as each step in the manufacturing process is completed.

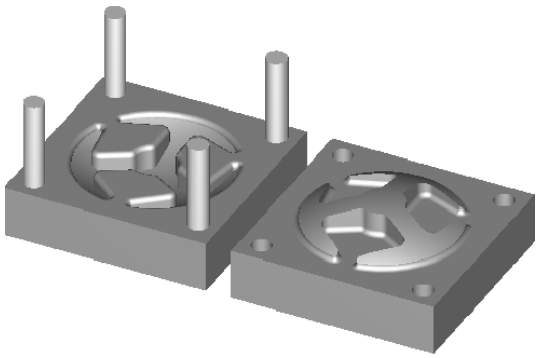


Figure 1.7: A die for die-casting.

The first phase of manufacturing processes is the design of a product. Computer-aided design (CAD) is a form of automation that helps designers prepare drawings, specifications, parts lists, and other design-related elements using special graphics and calculations intensive computer programs. Computer-aided design systems have considerably simplified industrial design, the first phase in the life of a new product.

Modeling. The first CAD system was the *Sketchpad* system [51], developed by Ivan Sutherland in the early 1960s. Although CAD systems originally automated 2-dimensional drawing and drafting, they now usually include 3-dimensional modeling and computer-simulated operation of the object. The history of modelings in CAD systems can be summarized as follows:

- *2-dimensional projections*: Entities (line, circle, arc and text) are projected on 2-dimensional planes. Several 2-dimensional views represent a 3-dimensional object. When you modify one of these drawings, you also need to change the others manually.
- *wire-frame*: The first modeling in 3-dimensional space. Edges are defined by lines or arcs connecting points in 3-dimensional space. This is the easiest and simplest way of representing a 3-dimensional model, but models need to be simple and clear. A 2-dimensional view of an object can easily be displayed.
- *surface modeling*: Surfaces interpolate edges of wire-frames. Curved surfaces can be represented with shading. Surface modeling is widely used where the quality of surfaces is important, for example, for the surfaces of the external bodies of cars and planes. It is impossible to extract physical properties of models. Figure 1.8 shows an example of surface modeling: an oilpump.
- *solid modeling*: A solid is a volume enclosed by surfaces that are represented as a quilt of vertices, edges, and faces. The major representations of solids include

constructive solid geometry, boundary representations, and spatial subdivision representations, all of which support the unambiguous, algorithmic determination of point membership: given any point $p = (x, y, z)$, there must be an algorithm that determines whether the point is inside, outside, or on the surface of the solid. Solid modeling maintains additional information on the interior and the exterior of the volume.

Solid modeling is in transition. As Hoffmann [27] writes, classical design paradigms that concentrated on obtaining one specific final shape are being supplanted by feature-based, constraint-based design paradigms that are oriented more toward the design process and define classes of shape instances. One of these new paradigms is parametric solid modeling which is a key technology to define and manipulate solid models through high-level, parameterized steps. A parametric solid can be defined as a solid whose actual shape is a function of a given set of parameters and constraints. The shape designer can define entire families of shapes, not just specific instances. Hoffmann's survey [28, 27] provides an excellent overview of solid modeling and parametric modeling.

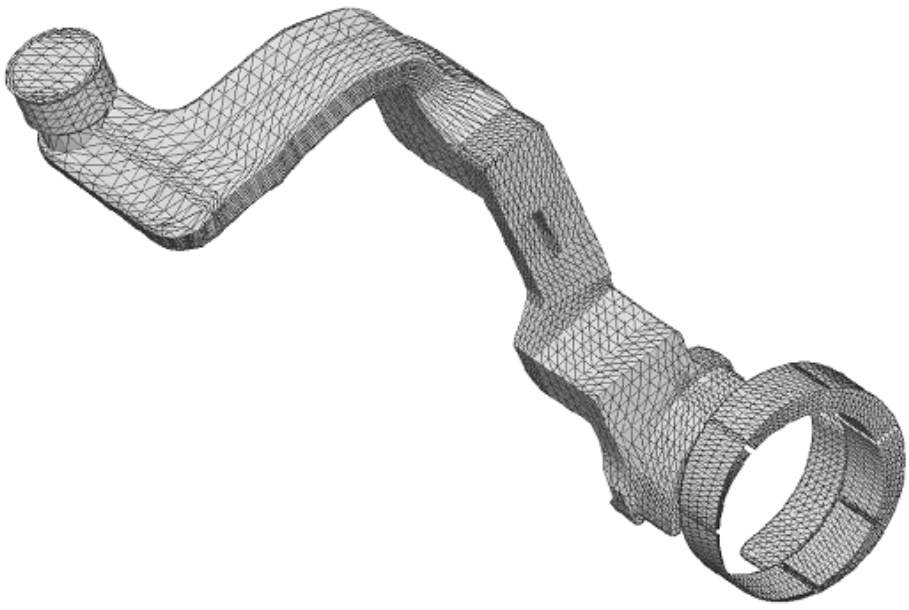


Figure 1.8: 3-dimensional surface modeling.

There are plenty of commercial CAD packages, such as AutoCAD™[1], UniGraphics™[2], SolidWorks™[3], Helix Design System™[4], SOLIDCAM™[5] and I-DEAS™[6], and most of them provide integrated features for surface modeling and solid modeling. Nowadays these packages have features to publish CAD drawings to the Web.

Verification and Simulation. Once an object has been designed, it has to be manufactured using the intended technique. As Bose and Toussaint [11, 14] write, it is desirable to design the object in such a way that manufacturing can be performed easily and cheaply. A fundamental question arises concerning every type of manufacturing process : Given an object, can it be constructed using a particular process?

The geometry of the object, coupled with the restrictions imposed by the particular manufacturing process under consideration, plays a vital role in determining the answer to the question. To answer the question, computer-aided design systems must be augmented with a component that verifies on-line whether an object being designed can actually be manufactured using the intended techniques long before the fabrication of costly physical models. Algorithms in such verification systems need to deduce the feasibility of manufacturing techniques purely on the basis of a CAD model of the object. Not only should they answer whether production is feasible, they can provide more information such as a list of possible orientations of the object that can build the object in the technique, a list of possible sequences of movements for manufacturing parts, and a simulation of the building process. In case the object is not feasible, they should point out what is wrong with the object.

Such algorithms have been proposed for a number of manufacturing processes, such as injection moulding [15, 21, 45], NC-machining [25], and stereolithography [10].

The importance of these verification components is quite evident. For example, when designing an object to be built using a certain technique, an engineer can check on-line whether the object can be built or not. By employing such components, computer-aided design systems help designers minimize scraps, reduce design time and eliminate wasted or redundant operations. These systems enable engineers to considerably reduce product-development costs and greatly shorten the design cycle.

1.4 Geometric aspects of the casting process

We now concentrate on the casting process. As discussed in the previous section, industrial computer-aided design systems could aid a part designer in verifying already during

[1] AutoCAD™ is a trademark of Autodesk, `http://www.autodesk.com`
[2] UniGraphics™ is a trademark of UGS, `http://www.ugs.com`
[3] SolidWorks™ is a trademark of SolidWorks Corporation, `http://www.solidworks.com`
[4] Helix Design System™ is a trademark of Microcadam Inc., `http://www.microcadam.com`
[5] SOLIDCAM™ is a trademark of CADTECH, `http://www.solidcam.com`
[6] I-DEAS™ is a trademark of SDRC, `http://www.i-deas.com`

the design of an object whether the object in question can actually be manufactured using a casting process. At the basis of this verification is a geometric decision: is it possible to enclose the object in a mould that can be split into two parts, such that these *cast parts* can be removed from the object without colliding with the object or each other? These geometric problems can generally be termed *separability* problems [53]. (We are not interested in casting processes where the mould has to be destroyed in order to remove the object, but only in situation where the given object can be mass-produced by re-using the same cast parts.) The casting process may fail in the removal of the cast parts: if the cast is not designed properly, then one or more of the cast parts may be stuck during the removal phase, as in Figure 1.9. The problems we address here concern this aspect: Given a 3-dimensional object, is there a cast for it whose parts can be removed after the liquid has solidified? An object for which this is the case is called *castable*. Note that this is a preliminary decision meant to aid in part design—to physically create the mould for a part one needs to take into account other factors such as heat flow and how air can evade from the cavity.

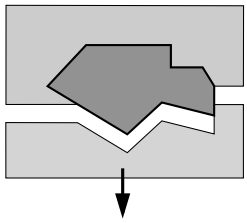


Figure 1.9: The top part of the cast is stuck.

In Chapter 3, 4, and 5, we consider the *castability problem* for three different casting models, give complete characterizations of castability in those models, and obtain algorithms to verify these conditions for polyhedral parts.

In manufacturing, features of an object imply manufacturing information that facilitates the process of analyzing manufacturability and the automated design of a cast for the object [44, 23]. Informally, features are a product's generic shapes or characteristics that are associated with engineering knowledge about the product [46, 47]. A small hole or a depression on the boundary of an object, for example, restricts the set of removal directions for which this object is castable, since the portion of the cast in the hole or in the depression must be removed from the object without breaking the object. Identifying such features not only facilitates the decision process, but also reduces the search space for castable directions. Features can also make the automated design of cast much easier. In Chapter 6 and 7, we define a geometric feature, the *cavity*, which is related to the castability of objects, and provide algorithms to extract it from objects.

Our approach is to extract the geometric essence of the object we designed and answer the question based on a purely geometric perspective. Most geometric components in commercial CAD packages serve as front-end processors. Once CAD models are created, geometric components import geometric data from CAD models, filter redundant edges, repair geometric and topological irregularities, and generate meshes.

1.5 Casting with opposite cast removal

In Chapter 3, we consider a casting model where the cast (mould) consists of two parts and these parts must be removed in opposite directions without damaging the parts or the object. This chapter is based on a paper with Mark de Berg, Prosenjit Bose, Siu-Wing Cheng, Dan Halperin, Jiří Matoušek, and Otfried Schwarzkopf [7].

Contrary to the *sand casting model* studied by Bose et al. [12] where the partition of the cast into two parts must be done by a plane, the cast is partitioned by a polygonal parting surface. In this casting model all convex polyhedra are castable. (In the sand casting model, even convex polyhedra are not always castable.)

In Section 3.2 we consider the case where the orientation of the object in the cast and the removal direction $\vec{d}$ are specified in advance. The problem then is to decide whether the object is castable in that direction, that is, whether the cast can be partitioned into two parts that can be removed in direction $\vec{d}$ and $-\vec{d}$, respectively. We give a necessary and sufficient condition under which such a partition exists : the object is *monotone* in the removal direction $\vec{d}$. In other words, an object $\mathcal{Q}$ is castable in direction $\vec{d}$ if and only if every line with direction $\vec{d}$ intersects the interior of $\mathcal{Q}$ in at most one connected component. The class of objects we allow in this characterization is more general than in previous works: the object need not be polyhedral and may have arbitrary genus. We give a simple way to *verify* the condition for polyhedral objects of arbitrary genus: a necessary and sufficient condition under which a polyhedron $\mathcal{P}$ is monotone in direction $\vec{d}$, and therefore castable in $\vec{d}$ is that $\mathcal{P}$ has no reflex silhouette elements and its shadow edges form a set of non-crossing curves. Silhouette elements are silhouette edges in $\vec{d}$, edges parallel to $\vec{d}$, and parts of facets parallel to $\vec{d}$. Shadow edges are the projection of convex silhouette elements onto a plane whose normal direction is parallel to $\vec{d}$. This condition leads to an $O(n \log n)$ time algorithm, where n is the combinatorial complexity of the polyhedron. We also give an algorithm that computes a partitioning of the cast into two removable parts (provided the polyhedron is castable, of course).

In Section 3.3 we consider the case where the removal direction is not specified in advance. Here the problem is to find all *combinatorially distinct directions* in which the object is castable (we postpone a formal definition of "distinct directions" to Section 3.3). One way of doing this is to generate a large set of sample directions, and to test each direction with the $O(n \log n)$ algorithm. This is the approach we take in the experimental section (Section 3.4) and it turns out to work well in practice. Such a sampling approach

is not complete, however: it might erroneously report that there are no good directions. Hence, in Section 3.3 we give an exact algorithm that computes all combinatorially distinct casting directions in $O(n^4)$ time: If we imagine the direction $\vec{d}$ changing continuously, there are events that may influence the castability of the object. These *critical events* can be represented by great circles and arcs of great circles on the unit sphere. These curves form an arrangement of complexity $\Theta(n^4)$ in the worst case. The algorithm makes use of the fact that the difference between two adjacent faces in the arrangement is quite small. It traverses the arrangement, and updates the castability information in constant time per edge involved, except that the computation at the starting point takes $O(n^2 \log n)$ time. We also show that there exist polyhedra for which there are $\Omega(n^4)$ combinatorially distinct casting directions. This implies that our algorithm is optimal in the worst case if we want to report all such directions.

1.6 Casting with directional uncertainty

The casting algorithms mentioned in the previous section assume perfect control of the casting machinery. When a cast part is removed, it is required that the part moves exactly in the specified direction. In practice, however, this will rarely be the case. As in all applications of robotics, we have to deal with imperfect control of the machinery, and a certain level of uncertainty in its movement. When a facet of the object or of a cast part is almost parallel to the direction in which the cast parts are being moved, then the two touching surfaces may damage each other when the mould is being opened. This can make the resulting object worthless, or it may wear away the surface of the mould so that it cannot be reused as often.

In Chapter 4, we consider a casting model that is identical to the model with opposite cast removal in Chapter 3, except that the cast machinery has a certain level of uncertainty in its directional movement: Given a removal direction $\vec{d}$ for a cast part, the part may move in a direction $\vec{d}'$ such that the angle between two directions is within a certain level of uncertainty. This chapter is based on a paper with Otfried Cheong and René van Oostrum [6].

In Section 4.3 we consider the case where the orientation of the object, the removal direction $\vec{d}$, and the level of directional uncertainty angle α are specified in advance. The problem then is to decide whether the object is castable in that direction with uncertainty α, that is, whether the cast can be partitioned into two parts that can be removed in any direction $\vec{d}'$ and $\vec{d}''$, respectively, such that the angles between $\vec{d}$ and $\vec{d}'$, and between $-\vec{d}$ and $\vec{d}''$ are smaller than or equal to α. We give a necessary and sufficient condition under which such a partition exists: the polyhedron is α-*monotone* and α-*safe*. We say that a polyhedron $\mathcal{P}$ is α-monotone in direction $\vec{d}$ for an angle α if $\mathcal{P}$ is monotone in direction $\vec{d}'$ for all directions $\vec{d}'$ with $\angle(\vec{d}, \vec{d}') \leq \alpha$. A polyhedron $\mathcal{P}$ is called α-safe in direction $\vec{d}$ if none of the normals of its facets make angles β with $\vec{d}$ in the range $\pi/2 - \alpha \leq \beta \leq \pi/2 + \alpha$.

The casting model we consider is more practical than the models in previous works, since most of the existing machinery bears a certain level of uncertainty. We give a simple way to verify the condition for polyhedral objects of arbitrary genus, leading to an $O(n \log n)$ time algorithm, where n is the combinatorial complexity of the polyhedron. We also give an algorithm that computes a partitioning of the cast into two removable parts.

In Section 4.4 we consider the case where the removal direction is not specified in advance. Depending on whether the uncertainty is specified in advance or not, we consider two problems. One of them is, for given uncertainty α, to find all *combinatorially distinct directions* in which the object is castable with directional uncertainty α. In Section 4.4 we give an exact algorithm that computes all combinatorially distinct casting directions in time $O(n^2 \log n/\alpha^2)$. We also consider an approximative solution, and give a heuristic that runs in time $O(n \log n)$ for constant α.

The other problem we consider is to find the *best* removal direction in which the object is castable. In this problem the *best* is qualified in the way that the directional uncertainty is as large as possible with which the object is castable. In Section 4.4 we give an exact algorithm that computes the best casting directions in $O(n^4)$ time. If it is known that $\mathcal{P}$ is α-castable for a certain angle α, we can compute the largest feasible uncertainty in time $O(n^2 \log n/\alpha^2)$. We also give a heuristic approach to approximate the largest feasible uncertainty.

1.7 Casting with skewed ejection direction

In most existing machinery, the retraction and ejection directions are identical as in Figure 1.5. Previous work on this problem has also assumed this restriction on casting. Existing technology for injection moulding, however, already has the flexibility to accommodate an ejection direction that is different from the retraction direction of the moving cast part. Exploiting this possibility allows to cast more parts, or to cast parts with simpler moulds. This is a generalization of the opposite casting model in the sense that the restriction on the removal directions of cast parts are removed.

In Chapter 5, we consider a casting model where the two cast parts are to be removed in two given directions and these directions need not be opposite. In contrast with previous works, the ordering of removal is important in this casting model. This chapter is based on a paper with Siu-Wing Cheng and Otfried Cheong [2].

We give a complete characterization of castability in this casting model, under the assumption that the cast has to consist of two parts that are to be removed in two not necessarily opposite directions. We also give an algorithm to verify this condition for polyhedral objects. We do not assume any special separability of the two cast parts, and allow parts of arbitrary genus. The running time of our algorithm for determining the castability of an object with a given pair of directions is $O(n^2 \log n)$.

All the results for opposite cast parts removal in [7, 30, 34] rely on the property that an object is castable if its boundary surface is completely visible from the two opposite removal directions. This is not true when the removal directions are non-opposite: there are polyhedra whose whole boundary is visible from the removal directions but which are not castable with respect to those directions [7].

For completeness, we also give an $O(n^{14}\log n)$-time algorithm for finding all combinatorially distinct feasible pairs of removal directions: we consider a 4-dimensional parameter space formed by the set of all pairs of directions, and construct a set of algebraic surfaces which correspond to a number of critical events that may influence the castability of the object. There are $O(n^3)$ surfaces and their arrangement has complexity $O(n^{12})$. We test at most $O(n^{12})$ pairs of directions using the algorithm of time complexity $O(n^2\log n)$ for determining the castability. Though the running time is polynomial, the algorithm is clearly of theoretical interest only.

1.8 The reflex-free hull

Computational geometers have defined many classes of 2-dimensional polygons, but few classes of 3-dimensional polyhedra. Perhaps the fact that 3-dimensional polyhedra support a rich class of topological structure in the form of knots and links has overshadowed the identification of geometric structure.

A small hole or depression on the boundary of an object, for example, restricts the set of directions for which this object is castable, since the portion of the cast in the hole or the depression must be removed without breaking the object. Most parts used in industry, such as engine rooms, telephone bodies, and small parts for cars and aircrafts, have such features.

This suggests a new approach to castability analysis: For a given pair of removal directions, we first identify such features (holes and depressions) of an object and if any such features cannot be accomodated with the given removal directions, then we can conclude that the object is not castable with the given removal directions. This idea can drastically reduce the size of the search space for feasible casting directions. For example, a hole with the shape of a cylinder in an object reduces the search space into a pair of two opposite directions parallel to the generators of the cylinder. Features, furthermore, can also be used for computing the minimum number of casting parts. In other words, the minimum number of additional casting parts (called *side cores*), together with two main parts, can be obtained from features.

In Chapter 6, we study features of a polyhedron related to casting, and define three geometric structures : *plane-cavities*, *cavities*, and the *reflex-free hull*. These definitions can also be applied to a 3-dimensional general shape. This chapter is based on work with Siu-Wing Cheng, Otfried Cheong, and Jack Snoeyink [4, 5].

In Section 6.3, we show several properties of the reflex-free hull of a polyhedron. One of the interesting properties of the reflex-free hull is that its complexity is linear in the size of the input polyhedron.

We currently have no algorithms for constructing the reflex-free hulls and cavities. In Chapter 7, we show how these geometric structures can be made use of with application to casting.

1.9 Coloring algorithm for finding cavities

Based on the definition of the reflex-free hull and cavities in Chapter 7, we consider applications of these geometric structures to casting. Cavities and the reflex-free hull are important features in applications such as manufacturing and molecular analysis. Unfortunately, we are currently unable to construct the reflex-free hulls and cavities. Nevertheless, we are able to prove that given a castable polyhedron, the bounding faces of a cavity necessarily belong to the same mould part. So we can make use of cavities in automatic mould part construction.

In Chapter 7, we present an algorithm to partition the faces of a polyhedron into disjoint subsets such that each subset must belong to the same mould part. Furthermore, we prove that the bounding faces of each cavity belong to the same subset. Thus, our algorithm is an effective method to restrict the search space for feasible casting directions. In fact, we conjecture that this algorithm can be extended so that, in the end, for any two distinct subsets, there is a feasible casting direction in which the mould is removed from the corresponding faces in opposite directions. This chapter is based on a paper with Siu-Wing Cheng, Otfried Cheong, and Jack Snoeyink [3].

CHAPTER 2

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 the chapter.

2.1 The model of the casting process

First we define the model of the casting process that is used in this thesis. In the real casting process, the movable part retracts from the fixed part carrying the object, after which the object is ejected from the retracted part as in Figure 1.5. To simplify our discussion, we will pretend that it is not the object that is ejected from the moving cast part, but that the cast part is removed from the object. In this way, we have symmetry between the two cast parts, and both retraction and ejection are modelled conceptually by removal of a cast part. To simulate the retraction, the fixed cast part will first be removed in a direction opposite to the retraction. Then to simulate the ejection, the remaining cast part will be removed in a direction opposite to the ejection. The directions in which the cast parts are removed are called the *removal directions* or *parting directions*. Figure 2.1 illustrates the process on a 2-dimensional example.

In our casting model, we assume that the cavity with the shape of the object is already filled with the molten material. We are only interested in the opening of the cast part without breaking the part and the object.

We assume that the outer shape of the cast $\mathcal{C}$ is the boundary of an axis-parallel box B, and we assume that B is large enough so that the object to be manufactured is contained in the interior of B. This assumption is necessary for producing connected cast parts. As stated in the introduction, we are interested in casts consisting of two parts. One of them will be called the *red cast part* and denoted by $\mathcal{C}_r$, the other will be called the *blue cast part* and denoted by $\mathcal{C}_b$. Two parts only overlap along their boundaries. Both $\mathcal{C}_r$ and $\mathcal{C}_b$

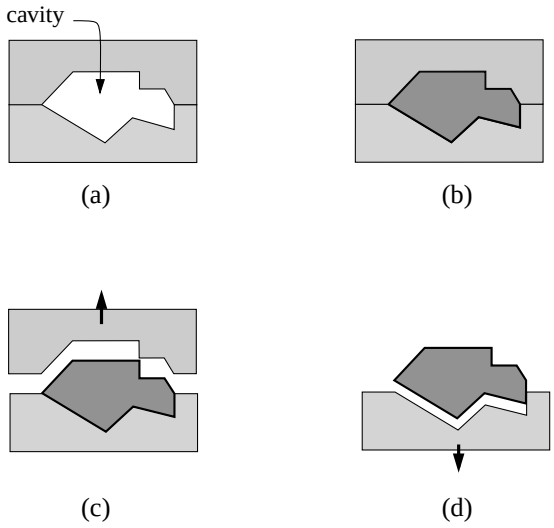


Figure 2.1: The casting process: simulation. (a) A cast formed by two parts (b) The cavity filled with the molten material (c) the retraction simulated (d) the ejection simulated

are connected subsets of B. The union of $\mathcal{C}_r$ and $\mathcal{C}_b$ equals $B \setminus \mathcal{Q}$, where $\mathcal{Q}$ is the object to be manufactured. The red cast part is removed in a certain direction that is called the *red removal direction*, denoted by $\vec{d}_r$, and the blue cast part is removed in the other removal direction called the *blue removal direction*, denoted by $\vec{d}_b$. In our casting model, we always remove the red cast part first, after which we remove the blue cast part.

2.2 Object models and other definitions

Throughout this thesis, $\mathcal{P}$ denotes a polyhedron, that is, a (not necessarily convex) solid bounded by a piecewise linear surface. The union of vertices, edges, and facets on this surface forms the boundary of $\mathcal{P}$, which we denote by $\partial\mathcal{P}$. We require that $\partial\mathcal{P}$ be a connected 2-manifold. Each facet of $\mathcal{P}$ is a connected planar polygon, which is allowed to have polygonal holes. Two facets of $\mathcal{P}$ are called *adjacent* if they share an edge. We assume that adjacent facets are not coplanar—coplanar facets should be merged into one—but we do allow coplanar non-adjacent facets. A polyhedron partitions the space into two disjoint domains, the *interior* and *exterior*. The open interior of the polyhedron $\mathcal{P}$ is denoted by $\text{int}(\mathcal{P})$, which is enclosed by $\partial\mathcal{P}$. The polyhedron consists of the boundary and its interior, that is, $\mathcal{P} = \text{int}(\mathcal{P}) \cup \partial\mathcal{P}$.

We also assume that $\mathcal{P}$ is *simple*, which means that no pair of non-adjacent facets shares a point. Our assumptions imply that $\mathcal{P}$ may contain tunnels, but no voids—a polyhedron

with a void is not castable anyway. As this thesis only deals with simple polyhedra, we will refer to them as polyhedra in the remainder of the thesis.

For a more thorough description of polyhedra and some of their properties, the reader is referred to the book by Preparata and Shamos [40].

A *convex edge* of a polyhedron $\mathcal{P}$ refers to an edge e where the dihedral angle through $\text{int}(\mathcal{P})$ of both facets sharing e is less than π. Similarly, a *reflex edge* refers to an edge e where the dihedral angle through $\text{int}(\mathcal{P})$ of both facets sharing e is greater than π.

Although we are primarily concerned with polyhedral sets, we give more general definitions and the characterizations of castability. In Chapter 3 and Chapter 5 we give characterizations of castability which apply to both polyhedra and curved objects, and in Chapter 6 we give definitions of geometric structures which apply to both polyhedra and curved objects. This is important since many industrial parts are not polyhedral. More precisely, we shall be dealing with objects $\mathcal{Q}$ whose bounding surface consists of a finite number of bounded-degree algebraic surface patches, which meet along bounded-degree algebraic curve segments. We further require that $\mathcal{Q}$ be topologically equivalent to a polyhedron $\mathcal{P}$ as defined above: it must be a solid whose boundary is a connected 2-manifold.

We denote the interior of an object $\mathcal{Q}$ by $\text{int}(\mathcal{Q})$, its closure by $\text{cl}(\mathcal{Q})$, and its boundary by $\partial\mathcal{Q}$. The projection of an object $\mathcal{Q}$ (usually the vertical projection onto the *xy*-plane) will be denoted by $\overline{\mathcal{Q}}$.

By a *direction* we mean an equivalence class of oriented parallel lines. A given direction $\vec{d}$ can be specified by a point on a unit sphere in the following way. On a unit sphere with center *o*, the origin, let *x* be a point on the boundary of the sphere such that the vector $\vec{ox}$ is parallel to and with the same orientation as $\vec{d}$. Then direction $\vec{d}$ is represented by the point *x*. (See Figure 2.2) A point that is diametrically opposite to *x* on the unit circle represents the *opposite* direction to $\vec{d}$ and is denoted by $-\vec{d}$.

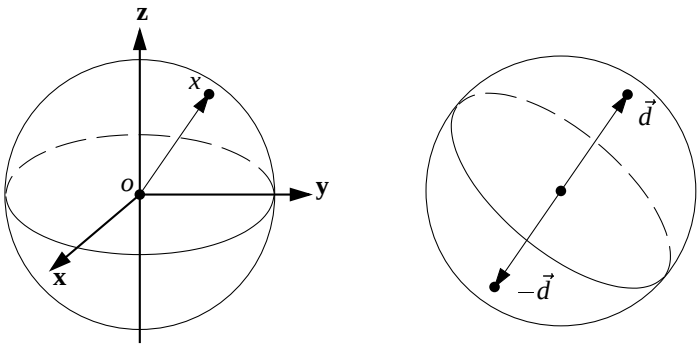


Figure 2.2: The sphere of directions

We call an object $\mathcal{Q}$ *$\vec{d}$-monotone* if every line with direction $\vec{d}$ intersects the interior of $\mathcal{Q}$ in at most one connected component. A *polyhedral terrain* is the graph of a (possibly partially defined) continuous piecewise linear function with domain $\mathbb{R}^2$ and range $\mathbb{R}$. This means that a polyhedral terrain is a polyhedral surface with the property that every vertical line intersects it in at most one point or segment. Hence, it is z-monotone.

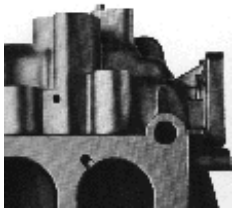
2.3 Arrangement

In our discussion we will refer to the subdivision of the unit sphere $\mathcal{S}^2$ induced by a collection $\mathcal{R}$ of great circles and arcs of great circles. We call this subdivision the *arrangement* of $\mathcal{R}$ on $\mathcal{S}^2$ and denote it by $\mathcal{A}(\mathcal{R})$. This arrangement consists of *faces* of dimensions $0, 1$, and 2, which are called vertices, edges, and cells, respectively. A vertex of $\mathcal{A}(\mathcal{R})$ is either an intersection point of two curves in $\mathcal{R}$ or an endpoint of an arc in $\mathcal{R}$. An edge of $\mathcal{A}(\mathcal{R})$ is a maximal connected component of a curve in $\mathcal{R}$ not intersecting any other curve in $\mathcal{R}$. A cell of $\mathcal{A}(\mathcal{R})$ is a maximal connected region of $\mathcal{S}^2$ not intersecting any curve in $\mathcal{R}$.

The *combinatorial complexity* of the arrangement $\mathcal{A}(\mathcal{R})$ is the total number of faces (of all dimensions) in the arrangement. If Q consists of n curves, each being a great circle or an arc of a great circle, then the complexity of the arrangement is $O(n^2)$ and there exists an arrangement whose complexity is $\Omega(n^2)$. We say that the curves in $\mathcal{R}$ are in *general position* if no three curves in $\mathcal{R}$ meet at a single point, and no two curves overlap in an arc of non-zero length (two curves are intersecting in at most two points).

Part II

Castability



CHAPTER 3

Opposite cast removal

3.1 Introduction

In this chapter we assume that the cast (mould) consists of two parts and these parts must be removed in opposite direction without damaging the parts or the object. In the example of Figure 3.1, the cast parts are removed in opposite directions. This need not always be possible; sometimes it may be necessary to remove them in non-opposite directions.

The casting process may fail in the removal of the cast parts: if the cast is not designed properly, then one or more of the cast parts may be stuck during the removal phase, as in Figure 3.2. The problem we address concerns this aspect: given a 3-dimensional object, is there a cast for it whose parts can be removed after the liquid has solidified? An object for which this is the case is called *castable*.

Clearly not every object is castable. The class of castable objects may be enlarged through the use of so-called cores and inserts [20, 41, 55]—appendages to the cast parts, which are removed after the liquid has hardened and before the cast parts themselves are removed. Cores and inserts allow the possibility of building more complicated objects. However, their use slows down the manufacturing process and makes it more costly. We do not consider the extra possibilities of cores and inserts. For the use of cores and inserts, you will find some information in Chapter 6 and Chapter 7.

Related work. The 2-dimensional version of our problem has been studied by Rappaport and Rosenbloom [42]. They presented an $O(n)$ time algorithm to determine whether a simple n-vertex polygon can be decomposed into two monotone chains, which is a sufficient and necessary condition for the polygon to be castable.

Hui and Tan [30] gave a heuristic approach to the 3-dimensional problem. Some candi-

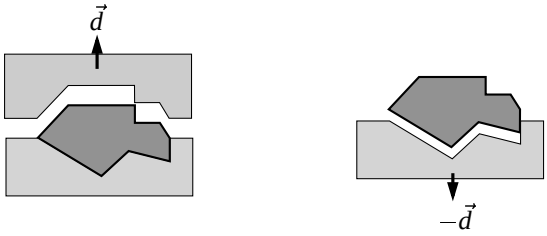


Figure 3.1: The cast parts are removed in opposite directions.

date removal directions are heuristically chosen, ordered, and tested. To test a candidate removal direction, every point in a sample set of points on the boundary of the object is checked to see if it can be removed in the given direction (or its opposite). If this is the case for each sample point, then the removal direction is assumed to be feasible. Kwong [34] gave an algorithm to determine the feasibility of a given parting direction.

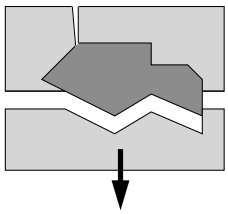


Figure 3.2: The top part of the cast is stuck.

He reduced the problem to the hidden surface removal problem in computer graphics by observing that if all the facets can be completely illuminated from the parting direction and its opposite, then the parting direction is feasible.

The algorithm of Chen et al. [17] first computes the convex hull of a polyhedral object, and then obtains the *pockets* of the object by subtracting it from its convex hull. Chen et al. observed that if all pockets are completely visible in either the parting direction or its opposite, then the parting direction is feasible. Their algorithm returns the parting direction that maximizes the number of completely visible pockets. However, as the converse of the above observation is not necessarily true, the algorithm is not complete and may not find a good parting direction even if one exists.

Hui [29] gave exponential time algorithms that also take cores and inserts into account. Since these algorithms are based on the work of Chen et al. they are not complete.

Finally, Bose et al. [12] considered a special model of casting, the *sand casting model*, where the partition of the cast into two parts must be done by a plane. Note that even convex polyhedra are not always castable in the sand casting model [12]. Bose et al. presented two algorithms for deciding whether for a given simple polyhedron with n vertices there is a cast whose constituent parts can be removed in opposite directions. One algorithm is based on partition trees [38] and uses $O(n^{3/2+\varepsilon})$ time and space,[1] the other is based on linear programming and uses $O(n^2)$ time and $O(n)$ space. When non-opposite directions are allowed, the running time of their partition-tree based algorithm remains $O(n^{3/2+\varepsilon})$, whereas the running time of their linear-programming based algorithm increases slightly to $O(n^2 \log n)$.

Summary of our results. This chapter is concerned with the case where the directions in which the two cast parts must be removed are opposite. For this case we obtain the following results.

In Section 3.2 we consider the case where the orientation of the object in the cast and the removal direction $\vec{d}$ are specified in advance. The problem then is to decide whether the object is castable in that direction, that is, whether the cast can be partitioned into two parts that can be removed in direction $\vec{d}$ and $-\vec{d}$, respectively. We give a necessary and sufficient condition under which such a partition exists. The class of objects we allow is more general than in previous works: the objects need not be polyhedral and they may have arbitrary genus. We give a simple way to verify the condition for polyhedral objects of arbitrary genus, leading to an $O(n \log n)$ time algorithm, where n is the combinatorial complexity of the polyhedron. We also give an algorithm that computes a partitioning of the cast into two removable parts (provided the polyhedron is castable, of course).

In Section 3.3 we consider the case where the removal direction is not specified in advance. Here the problem is to find all *combinatorially distinct directions* in which the object is castable (we postpone a formal definition of "distinct directions" to Section 3.3). One way of doing this is to generate a large set of sample directions, and test each direction with the $O(n \log n)$ algorithm. This is the approach we take in the experimental section (Section 3.4) and it turns out to work well in practice. Such a sampling approach is not complete, however: it might erroneously report that there are no good directions. Hence, in Section 3.3 we give an exact algorithm that computes all combinatorially distinct casting directions in $O(n^4)$ time. We also show that there exist polyhedra for which there are $\Omega(n^4)$ combinatorially distinct casting directions. This implies that our algorithm is optimal in the worst case if we want to report all such directions.

[1] Bose et al. remarked that this can be improved to $O(n^{4/3+\varepsilon})$. The parameter ε in these bounds is a positive constant, which can be chosen arbitrarily small.

3.2 Testing a direction

In this section we present a criterion for testing whether a given object $\mathcal{Q}$ admits opposite cast removal in a given direction $\vec{d}$. In other words, we give a way to determine whether a cast $\mathcal{C}$ for $\mathcal{Q}$ can be split into a red part $\mathcal{C}_r$ and a blue part $\mathcal{C}_b$ that can be translated to infinity in direction $\vec{d}$ and $-\vec{d}$, respectively, so that the interior of $\mathcal{C}_r$, $\mathcal{C}_b$, and $\mathcal{Q}$ do not intersect during the translations. If this is the case, we say that $\mathcal{C}_r$ and $\mathcal{C}_b$ can be removed *without collision* and $\mathcal{Q}$ is *castable in direction* $\vec{d}$. The order of removing the cast parts is irrelevant in this situation.

Throughout this section, and without loss of generality, we assume that $\vec{d}$ is the vertical direction—the positive z direction—and we say that $\mathcal{Q}$ is castable if it is castable in the vertical direction. The red cast part has to be translated upward, the blue cast part downward.

Lemma 1 *An object $\mathcal{Q}$ is castable if and only if it is vertically monotone.*

Proof: Assume that $\mathcal{Q}$ is castable, and let $\mathcal{C} = (\mathcal{C}_r, \mathcal{C}_b)$ be a two-part cast whose parts are removable. Let ℓ be a vertical line intersecting $\mathcal{Q}$, and let p and q be two points in $\ell \cap \text{int}(\mathcal{Q})$. Since a point $r \in \ell$ in between p and q can be translated neither upward nor downward without colliding with $\mathcal{Q}$, the point r can be in neither $\mathcal{C}_r$ nor $\mathcal{C}_b$. Hence $r \in \mathcal{Q}$. We must even have $r \in \text{int}(\mathcal{Q})$; otherwise there would be a point $r' \notin \mathcal{Q}$ having a point $p' \in \mathcal{Q}$ above it and a point $q' \in \mathcal{Q}$ below it—this follows from p and q being in the interior of $\mathcal{Q}$—and such a point r' can be in neither $\mathcal{C}_r$ nor $\mathcal{C}_b$. This proves that $\mathcal{Q}$ is vertically monotone.

Assume now that $\mathcal{Q}$ is vertically monotone, and recall that the cast $\mathcal{C}$ is made from a rectangular axis-parallel box B. Let $\mathcal{Q}^*$ be the solid obtained by sweeping $\mathcal{Q}$ upward to infinity. We let $\mathcal{C}_r := (\mathcal{Q}^* \cap B) \setminus \mathcal{Q}$ be the red cast part, and we let $\mathcal{C}_b := B \setminus (\mathcal{Q} \cup \mathcal{C}_r)$ be the blue cast part. Because $\mathcal{Q}$ is vertically monotone, $\mathcal{C}_r$ is connected and can be translated upward to infinity without intersecting the interior of $\mathcal{Q}$, and without colliding with $\mathcal{C}_b$. Since any point above $\mathcal{Q}$ lies in $\mathcal{C}_r$ by definition, $\mathcal{C}_b$ can be translated downward without colliding with $\mathcal{Q}$. Because $\mathcal{C}_b$ is connected, we have constructed a two-part cast whose constituent parts can be removed. Hence, $\mathcal{Q}$ is castable. □

Let's turn our attention to the special case of a polyhedral object $\mathcal{P}$. The red and blue cast parts induce a partition of $\partial\mathcal{P}$ into a red part and a blue part. We call a facet of $\mathcal{P}$ an *up-facet* if its outward normal points upward (that is, has a positive z-component), and a *down-facet* if its outward normal points downward (that is, has a negative z-component). Vertical facets are neither up- nor down-facets. If $\mathcal{P}$ is castable, then clearly every up-facet must be completely red, while every down-facet must be blue. The object shown in Figures 3.3 illustrates that vertical facets sometimes need to be colored partly red and partly blue. In the figure, the facet *abcd* is coplanar with the vertical facets incident to

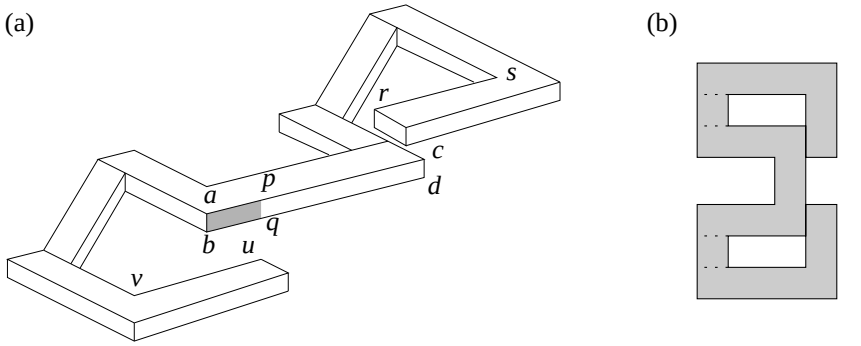


Figure 3.3: A polyhedron with a facet that needs two different colors. (a) A 3D view of the polyhedron. (b) The projection onto the *xy*-plane of the polyhedron, with the interior shaded.

the edges *uv* and *rs*. The line segment *pq* is the intersection of *abcd* with a vertical line through *u*. The construction in the proof of Lemma 1 colors all vertical facets blue, except for the facet *abcd* which is partly blue (namely the part *pqcd*) and partly red (namely the part *abpq*). The fact that *abcd* receives two colors is not an artifact of the proof: any legal cast for this object in the vertical direction will assign two colors to *abcd*.

The construction used in Lemma 1 does not result in practically useful casts, since it generates many vertical walls between the red and blue cast parts. Sometimes these are unavoidable, as for the object in Figure 3.3, but it would be preferable to have a method that does not create any vertical walls if they are not necessary. We present such a method.

Theorem 1 *Let $\mathcal{P}$ be a vertically monotone polyhedron with n vertices. It is possible to construct a cast for $\mathcal{P}$ in $O(n \log n)$ time, such that the two cast parts do not meet along vertical facets if no vertical line avoiding the interior of $\mathcal{P}$ touches two non-adjacent facets of $\mathcal{P}$.*

Proof: Let h be a plane that is parallel to the *xy*-plane and cuts the box B into two halves. Let R be the rectangle $h \cap B$. We project all up-facets of $\mathcal{P}$ onto h and obtain a polygon $\overline{\mathcal{P}}$ with holes. Figure 3.4 shows the polygon we get when we project the polyhedron of Figure 3.3. Note that collinear edges are not merged in the projection, so that every vertex of an up-facet that projects onto the boundary of $\overline{\mathcal{P}}$ actually gives rise to a vertex of $\overline{\mathcal{P}}$. To compute a description of this polygon (in the form of a doubly-connected edge list [19], for instance), we need to determine the union of the projection of the up-facets. This can be done in $O(n \log n)$ time with a plane sweep algorithm (see Preparata and Shamos [40] for details on plane sweep).

Every point on $\partial\overline{\mathcal{P}}$ is the projection of a point on $\partial\mathcal{P}$ and, in fact, every vertex of $\overline{\mathcal{P}}$ is the

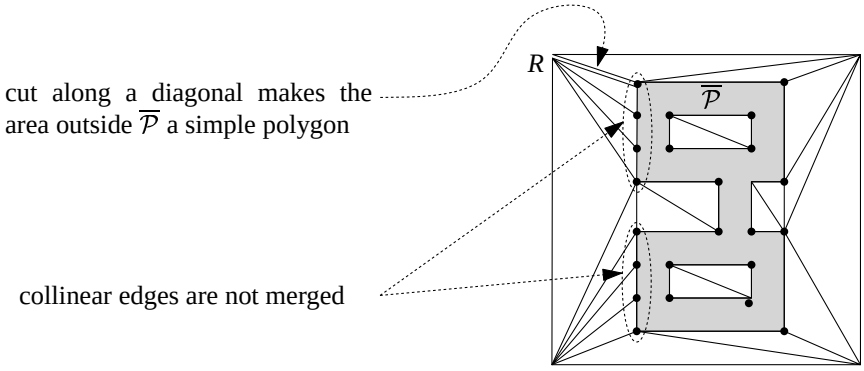


Figure 3.4: The projection of the polyhedron of Figure 3.3, and a triangulation of its complement.

projection of a vertex of $\mathcal{P}$. With each vertex $\bar{v}$ of $\overline{\mathcal{P}}$, we can therefore associate a vertex v of $\mathcal{P}$ that projects onto $\bar{v}$. If there is more than one such vertex, we choose the one with largest z-coordinate.

Since every vertex of $\overline{\mathcal{P}}$ is the projection of a vertex of $\mathcal{P}$, the complexity of $\overline{\mathcal{P}}$ is $O(n)$. Each of the holes of $\overline{\mathcal{P}}$ is a simple polygon, which can be triangulated in linear time [16]. The same is true for the part of the complement "outside" $\overline{\mathcal{P}}$. (This is not a simple polygon, but it can be made into one by cutting it open along a diagonal from a vertex of $\overline{\mathcal{P}}$ to a vertex of R, as is illustrated in Figure 3.4.) Hence, we obtain a triangulation $\overline{\sigma}$ of the complement of $\overline{\mathcal{P}}$ in $O(n)$ time. Every triangle of this triangulation is now "lifted" into 3-dimensional space by replacing every vertex $\bar{v}$ by its associated vertex v. We obtain a triangulated surface σ inside the box B. The surface σ defines the partition of the cast into two parts, as explained next.

First, let's assume that no vertical line avoiding the interior of $\mathcal{P}$ touches two non-adjacent facets of $\mathcal{P}$. Consider a triangle $\bar{t} \in \overline{\sigma}$ that shares an edge $\bar{e}$ with $\overline{\mathcal{P}}$. This edge is the projection of a unique edge e of an up-facet f of $\mathcal{P}$, and the lifted version of $\bar{t}$ will share e with f. This implies that the union of σ and the up-facets of $\mathcal{P}$ is a continuous surface σ^*. We let $\mathcal{C}_r$ be the part of B above σ^*, and we let $\mathcal{C}_b$ be the part of $B \setminus \mathcal{P}$ below σ^*. Note that every vertical line intersecting B will intersect σ^* in exactly one point. This implies that $\mathcal{C}_r$ and $\mathcal{C}_b$ do not meet along vertical facets. Together with the monotonicity of $\mathcal{P}$ it also implies that $\mathcal{C}_r$ can be removed upward and that $\mathcal{C}_b$ can be removed downward.

Now consider the general case, where there can be vertical lines that avoid the interior of $\mathcal{P}$ but touch two or more non-adjacent facets of $\mathcal{P}$. In this case we still let $\mathcal{C}_r$ be the part of B above σ^*. However, σ^* is not continuous anymore. Figure 3.5 illustrates this for our running example. In this figure, the up-facets are darkly shaded and σ is lightly shaded; for clarity, some of the triangles in σ—the ones lying more in the back—have

been omitted. To make σ^* into a continuous surface we have to add certain vertical

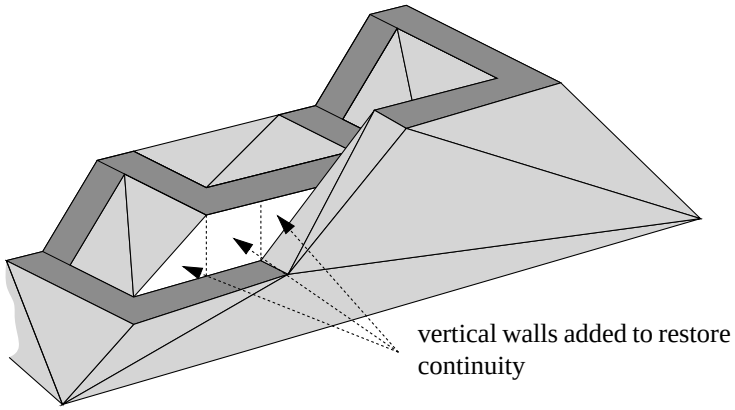


Figure 3.5: A discontinuity in the surface σ^*.

walls at places where a vertical line avoiding the interior of $\mathcal{P}$ touches two or more non-adjacent facets of $\mathcal{P}$. More precisely, we need to add the following vertical walls. If edges of two different up-facets overlap in the projection, then we need a rectangular vertical wall to connect the portions of these edges that overlap in the projection—see the middle vertical wall added in Figure 3.5. Furthermore, if a triangle $\bar{t}$ shares an edge $\bar{e}$ with a projected up-facet but the lifted version t does not have e as an edge because it was lifted to a different height, then t needs to be connected to e with one or two triangular vertical walls—see the left and right vertical walls in Figure 3.5. After adding these vertical walls, σ^* is a continuous surface with the property that its intersection with any vertical line is connected. Together with the monotonicity of $\mathcal{P}$ this implies that both $\mathcal{C}_r$, the part of B above σ^*, and $\mathcal{C}_r$, the part of $B \setminus \mathcal{P}$ below σ^*, are removable. □

To check a polyhedron $\mathcal{P}$ for castability, we can use Lemma 1 and the following simple observation.

Observation 1 *A polyhedron $\mathcal{P}$ is vertically monotone if and only if the union of open up-facets forms a terrain.*

To test if the union of open up-facets forms a terrain, we can project them onto the *xy*-plane and check whether any pair intersects. This observation is essentially the basis for Kwong's algorithm [34]—see Section 3.1—and immediately implies Theorem 3. However, we elaborate in some detail a somewhat different approach that decides monotonicity by only looking at silhouette edges (which we will define below) of the polyhedron; the main reason being that silhouette edges can be updated efficiently when a direction change

occurs. Thus, using silhouette edges increases the efficiency of the algorithm presented in the next section, that goes over all possible directions in order to report the directions where a given polyhedron is castable.

We first give a precise definition of the silhouette of an object. This turns out to be somewhat tricky if the object has vertical facets.

Let $\mathcal{Q}$ be an object, and consider a vertical line ℓ. The line ℓ intersects $\partial\mathcal{Q}$ in a number of maximal closed intervals. These intervals separate open intervals lying either completely inside or outside $\mathcal{Q}$. A boundary interval that is surrounded on both sides by intervals outside $\mathcal{Q}$ is called a *convex silhouette interval*. A boundary interval that is surrounded on both sides by intervals in the interior of $\mathcal{Q}$ is called a *reflex silhouette interval*. The union over all vertical lines of all convex silhouette intervals forms the *convex silhouette* and the union of all reflex silhouette intervals forms the *reflex silhouette*. The union of the convex and reflex silhouettes is called the *silhouette* of $\mathcal{Q}$.

We visualize these definitions for the case of a polyhedron $\mathcal{P}$. If $\mathcal{P}$ has no vertical facets, then the silhouette consists exactly of the *silhouette edges* of $\mathcal{P}$, namely the edges e where the two facets incident to e lie on one side of the unique vertical plane through e. A silhouette edge is convex if the dihedral angle between the two facets in the interior of $\mathcal{P}$ is smaller than π, otherwise it is reflex. If $\mathcal{P}$ has vertical facets, then the silhouette is no longer 1-dimensional. For instance, the silhouette of the object in Figure 3.3 consists of *all* vertical facets of the polyhedron. Figure 3.6 shows another object with part of its quite complicated silhouette shown shaded. Note that the segments a and b are also part of the silhouette.

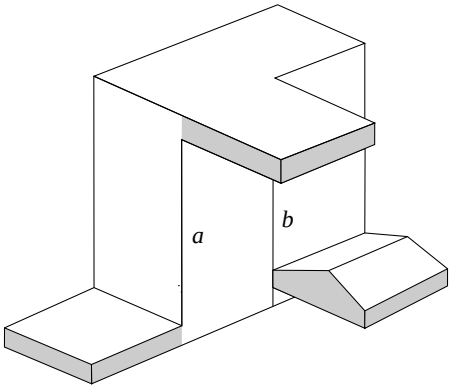


Figure 3.6: The silhouette of a polyhedron with vertical facets.

The following lemma is straightforward:

Lemma 2 *If the reflex silhouette of an object $\mathcal{Q}$ is not empty, then $\mathcal{Q}$ is not vertically monotone.*

From now on, we will therefore only consider objects whose reflex silhouette is empty. The silhouette is the convex silhouette in this case, and it consists of a finite number of disjoint curves or "bands" on $\partial\mathcal{Q}$, referred to as the *silhouette curves*. When the silhouette curves are projected vertically onto the *xy*-plane, we get a collection of so-called *shadow curves* in the plane. These are 1-dimensional curves, since the bands of the silhouette are vertical. Furthermore, the shadow curves are closed curves.

The key step in our argument is the following lemma.

Lemma 3 *Let $\mathcal{Q}$ be an object with empty reflex silhouette and such that no vertical line contains two silhouette intervals. Then $\mathcal{Q}$ is vertically monotone.*

Proof: Let S be the silhouette of $\mathcal{Q}$. Since no vertical line contains two silhouette intervals, the shadow curves of $\mathcal{Q}$ are a collection of mutually disjoint, simple, closed curves $\gamma_1, \gamma_2, \ldots, \gamma_k$ in the *xy*-plane that partition the plane into open regions R_1, R_2, ..., R_{k+1}, every one of which is topologically equivalent to a disc with a finite number of holes, as in Figure 3.7. If there is only one curve, the lemma holds trivially.

If there is more than one curve, then one of the curves, say γ_1, must contain all the other curves in its interior. Call this curve the *outer* curve. Let R_e be a region containing no holes bounded by a curve γ_e that is not the outer curve. Such a region must exist since all the curves are disjoint. Let γ_e separate R_e from R_f.

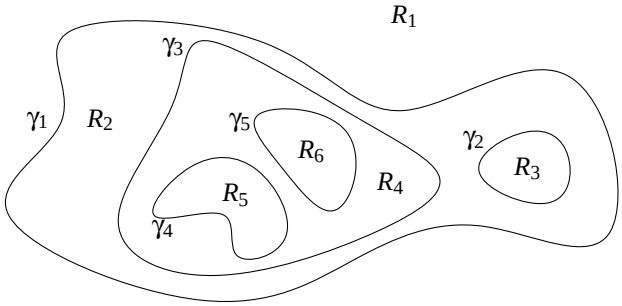


Figure 3.7: A collection of shadow curves, and the regions they define.

For a point p in the *xy*-plane, let $\ell(p)$ denote the vertical line through p. The set $\ell(p) \cap \mathcal{Q}$ is a disjoint union of closed intervals. We number those intervals from bottom to top as $I_1(p)$, $I_2(p)$, ..., $I_m(p)$. Within each region, the number m is a constant. Between two regions separated by a curve, the number m differs by one, since a curve introduces a

new interval in one of the regions. Let $I_d(p)$ be the interval introduced by the curve γ_e in region R_e. Consider the component $C = \bigcup_{p \in R_e} I_d(p)$. Since the silhouette curves are convex, there is no path from a point $p \in \mathcal{Q} \setminus C$ to a point in C. However, this violates the fact that $\mathcal{Q}$ is connected. Therefore, there can only be one curve, implying that $\mathcal{Q}$ is vertically monotone. □

The condition in the lemma above is sufficient for an object to be vertically monotone, but not necessary. This can be seen in Figure 3.3, which depicts a vertically monotone object for which there is a vertical line containing two silhouette intervals. That the condition is not satisfied can also be seen from the fact that the shadow curve is not simple—see Figure 3.3(b). To extend the lemma to a more general setting, we have to define clearly what kind of non-simplicity we allow. To this end we orient all silhouette curves such that the interior of $\mathcal{Q}$ lies locally to the left of the curve. This induces an orientation in the shadow curves, so that 'to the left of a shadow curve' is well defined. We say that a set of shadow curves is *non-crossing* if a slight shrinking of every curve—obtained by moving every point slightly to the left—results in a set of mutually disjoint, simple curves.

Figure 3.8: Shrinking the shadow curve of Figure 3.3(b).

We can now prove the generalization of the previous lemma.

Lemma 4 *Let $\mathcal{Q}$ be an object with empty reflex silhouette and non-crossing shadow curves. Then $\mathcal{Q}$ is vertically monotone.*

Proof: Assume that $\mathcal{Q}$ is not vertically monotone. Then there is a vertical line ℓ containing two points p and q in the interior of $\mathcal{Q}$, and a point r between p and q outside or on the boundary of $\mathcal{Q}$. Let $\varepsilon > 0$ be such that the balls of radius 2ε centered at p and q are contained in $\mathcal{Q}$. We shrink $\mathcal{Q}$ to obtain a new object $\mathcal{Q}'$ by removing from $\mathcal{Q}$ every point that has distance at most ε to the complement of $\mathcal{Q}$. If ε is chosen small enough, this results in a legal object $\mathcal{Q}'$, that is, a solid whose boundary is a connected 2-manifold. Since p and q lie in the interior of $\mathcal{Q}'$ and r lies outside $\mathcal{Q}'$ and between p and q, the object $\mathcal{Q}'$ is not vertically monotone. However, if the silhouette of $\mathcal{Q}$ consists of non-crossing convex silhoutte curves, then the silhouette of $\mathcal{Q}'$ consists of disjoint convex

silhoutte curves because of the shrinking. But then Lemma 3 implies that $\mathcal{Q}'$ is vertically monotone, a contradiction. □

The lemmas above give a sufficient condition for an object to be vertically monotone. The next lemma shows that the condition is necessary.

Lemma 5 *Let $\mathcal{Q}$ be an object with empty reflex silhouette whose shadow curves are crossing. Then $\mathcal{Q}$ is not vertically monotone.*

Proof: Let $\overline{p}$ be a point in the xy-plane where a shadow curve crosses itself or another shadow curve. A vertical line through $\overline{p}$ touches the boundary of $\mathcal{Q}$ in two points p and p' lying in two different silhouette intervals. This implies the existence of a point r between p and p' that lies outside $\mathcal{Q}$. A slight perturbation of this line shows that there exists a point outside $\mathcal{Q}$ lying between two points in the interior of $\mathcal{Q}$. Therefore, $\mathcal{Q}$ is not vertically monotone. □

We summarize Lemmas 2, 4, and 5 in the following theorem.

Theorem 2 *An object $\mathcal{Q}$ is vertically monotone, and therefore castable, if and only if its reflex silhouette is empty and its shadow curves are non-crossing.*

Before we outline our algorithm for testing castability, we have to examine the silhouette of a polyhedron $\mathcal{P}$ in more detail. This silhouette consists of silhouette edges, vertical edges of the polyhedron, and parts of vertical facets. To find the silhouette on the vertical facets correctly, we use the vertical decomposition of these facets. Every trapezoid Δ of the decomposition is bounded from above and from below by (parts of) edges e_1 and e_2 of $\mathcal{P}$. If both e_1 and e_2 are convex edges, then Δ is part of the convex silhouette of $\mathcal{P}$. If both e_1 and e_2 are reflex edges, then Δ is part of the reflex silhouette, and $\mathcal{P}$ is not castable. If one edge is convex while the other is reflex, Δ does not belong to the silhouette. Note that certain vertical extensions produced by the vertical decomposition also belong to the silhouette. We can ignore them, however, as their projection coincides with the projection of the endpoints of the incident non-vertical edges of trapezoids. We call the projection of every silhouette edge and silhouette trapezoid a *shadow edge*. By Theorem 2, the polyhedron $\mathcal{P}$ is castable if and only if it has no reflex silhouette elements, and its shadow edges form a set of non-crossing curves.

To decide on castability, we have to be able to test whether two shadow curves cross. If we examine the possible intersections of two shadow edges e_i and e_j, we find that there are four cases that have to be treated as crossings—see Figure 3.9. The four cases are as follows:

(i) the interiors of e_i and e_j intersect;

(ii) e_i and e_j overlap and have the same orientation;

(iii) an endpoint of e_i lies on e_j, and e_i lies to the left of e_j;

(iv) an endpoint of e_i coincides with the destination of the directed edge e_j, and e_i is contained in the wedge formed by e_j and the next shadow edge e_k on the shadow curve of e_j.

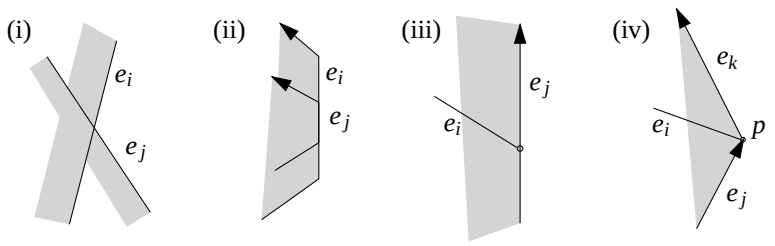


Figure 3.9: Four different ways in which shadow curves can cross.

Note that the condition on the orientation in (ii) ensures that the shadow curve in Figure 3.8 is correctly labeled as non-crossing.

By combining Theorem 2 with the characterization of crossing edges, we can compute efficiently whether a polyhedron is castable.

Theorem 3 *Given a polyhedron $\mathcal{P}$ with n vertices, we can test in time $O(n \log n)$ whether $\mathcal{P}$ is vertically monotone and therefore castable.*

Proof: We first form the vertical decomposition of all vertical facets in $O(n \log n)$ time. Next we identify all silhouette curves; a local analysis of all edges and trapezoids in the vertical decomposition of the vertical facets suffices for this. If there are any reflex silhouette intervals we can stop and report the polyhedron to be non-castable. Otherwise, we project the silhouette elements onto the *xy*-plane to get the shadow edges. A simple $O(n \log n)$ time plane sweep algorithm can then be used to determine whether there is a crossing in the collection of shadow curves. If we detect a crossing, we stop and report the polyhedron non-castable, and if the plane sweep proceeds without finding a crossing then the polyhedron is reported castable. □

3.3 Finding a direction

We have seen how to test whether a polyhedron $\mathcal{P}$ is castable in a given direction $\vec{d}$. In this section we describe an algorithm to solve the following problem: Given a polyhedron $\mathcal{P}$,

decide whether it can be cast in some direction $\vec{d}$. In fact, we will solve the more general problem of finding all directions $\vec{d}$ for which $\mathcal{P}$ can be cast.

We represent every possible direction by a point on the unit sphere $\mathcal{S}^2$ (centered at the origin): a point p on $\mathcal{S}^2$ represents the direction $\vec{d}_p$ from the origin to p. Our goal is to identify the region of $\mathcal{S}^2$ corresponding to directions in which $\mathcal{P}$ is castable. By Lemma 1 and Observation 1 $\mathcal{P}$ is castable in direction $\vec{d}$ if and only if the union of open up-facets forms a terrain relative to $\vec{d}$.

If we imagine the direction $\vec{d}$ changing continuously, there are two events that may influence the terrain-property of the up-facets: First, an up-facet may become a down-facet, or vice versa—the set of these directions forms $O(n)$ great circles on $\mathcal{S}^2$. Second, the projection of a vertex v of the polyhedron $\mathcal{P}$ may cross the projection of an edge e of $\mathcal{P}$—the set of these directions can be described by $O(n^2)$ arcs of great circles. Let $\mathcal{Q}$ denote the union of these curves which represent "critical events."

Consider the arrangement $\mathcal{A}(\mathcal{Q})$ of $O(n^2)$ great circles and great circle arcs on $\mathcal{S}^2$. Recall that the arrangement consists of faces of dimensions 0, 1, and 2, which we refer to as vertices, edges and cells respectively. For simplicity of exposition we first assume that the arrangement $\mathcal{A}(\mathcal{Q})$ is in *general position* (see Section 2.3), and later relax this assumption.

It is easily verified that inside every face of the arrangement $\mathcal{A}(\mathcal{Q})$, the polyhedron $\mathcal{P}$ is either $\vec{d}$-monotone for every direction $\vec{d}$, or for none. We say that two directions are *combinatorially distinct* if they lie in two different faces of the arrangement. We aim to compute all combinatorially distinct directions in which $\mathcal{P}$ is castable.

By our definition of castability, if a cell or an edge of $\mathcal{A}(\mathcal{Q})$ represents directions in which $\mathcal{P}$ is castable, then any vertex on its boundary represents a direction in which $\mathcal{P}$ is castable. This suggests the following simple algorithm. We compute all the vertices of the arrangement $\mathcal{A}(\mathcal{Q})$ by computing the intersections of all pairs of curves in $\mathcal{Q}$. This takes $O(n^4)$ time. Then for each vertex (intersection vertices and arcs endpoints) we test in $O(n \log n)$ time whether $\mathcal{P}$ is castable in the corresponding direction using Observation 1 and Theorem 3. The total running time of this algorithm is $O(n^5 \log n)$.

There are several ways in which this straightforward approach can be improved. Note that any vertex v in $\mathcal{A}(\mathcal{Q})$ represents a degenerate casting direction $\vec{d}_v$: either one facet of $\mathcal{P}$ or more are parallel to $\vec{d}_v$, or a line parallel to $\vec{d}_v$ intersects $\mathcal{P}$ in more than one connected component. Therefore, we may be better off proposing directions in the interior of cells of $\mathcal{A}(\mathcal{Q})$, if such exist. However, we need to consider all faces of $\mathcal{A}(\mathcal{Q})$ instead of just cells, because there may be directions of castability that appear only along edges (such as the situation depicted in Figure 3.3) or vertices of the arrangement.

Note also that the difference between two adjacent faces in $\mathcal{A}(\mathcal{Q})$ is quite small, provided that $\mathcal{A}(\mathcal{Q})$ is in general position. When going from one face of the arrangement to another, either one facet of $\mathcal{P}$ changes its status (among down-facet, up-facet or parallel relative to a given direction) or the projection of a vertex of $\mathcal{P}$ crosses into (or over) the projection

of an edge of $\mathcal{P}$.

To exploit the coherence between adjacent faces we proceed as follows. First we compute the arrangement $\mathcal{A}(\mathcal{Q})$. We do this with an output-sensitive algorithm. Let m denote the combinatorial complexity of $\mathcal{A}(\mathcal{Q})$. The algorithm is a straightforward adaptation of the plane-sweep paradigm to the sphere and its running time is $O((n^2+m)\log n)$. (We could also use here a randomized incremental construction algorithm whose expected running time is $O(n^2\log n+m)$; see, e.g., [39].) Its output is a data structure that allows for a traversal of the arrangement face by adjacent face (we could use, say, the quad-edge data structure for the purpose [22]).

If we are only concerned with worst-case running time, and since the complexity of the arrangement $\mathcal{A}(\mathcal{Q})$ can be $\Theta(n^4)$ in the worst case (see below), we can compute the arrangement in $\Theta(n^4)$ time by substituting each arc in $\mathcal{Q}$ by the great circle containing it and computing the arrangement of the resulting collection of great circles. (The latter arrangement is a refinement of $\mathcal{A}(\mathcal{Q})$ from which we could easily obtain the required output.) Using central projection from the sphere onto two parallel planes tangent to $\mathcal{S}^2$ at two antipodal points we obtain two arrangements of straight lines. Such arrangements can be computed in $\Theta(n^4)$ time each. In fact computing the arrangement on one plane suffices since the two arrangements are symmetric, provided some caution is exercised in choosing the projection planes: we choose a tangent plane π such that the great circle γ parallel to π does not fully contain a curve of $\mathcal{Q}$, and such that no vertex of the arrangement lies on γ. This way no information is lost by the projection.

After $\mathcal{A}(\mathcal{Q})$ has been computed, we choose a point p inside a face of the arrangement arbitrarily and compute the silhouette elements of $\mathcal{P}$ corresponding to the direction $\vec{d}_p$. For each silhouette element we check whether it is convex or reflex. We initialize two counters: how many silhouette elements are reflex and how many pairs of shadow edges cross one another. For the direction $\vec{d}_p$, this computation takes $O(n^2\log n)$ time using a plane-sweep algorithm. By Theorem 2, the polyhedron is castable in direction $\vec{d}_p$ if both counters are zero. We move to an adjacent face of the arrangement—we traverse the arrangement in, say, depth-first order on the graph induced by the edges and vertices of the arrangement. By looking at the edges of the arrangement that are involved in the move, we know how to update the counters at constant time per edge involved. At the end of the move we check the counters and report castability if they are both zero. If we report castability at a vertex, we also report castability at its incident edges and faces (in general this is only true under the general position assumption). Thus, after the computation at the starting point p, the entire traversal of the arrangement takes time $O(m)$. We conclude that all directions for which there is a good cast can be computed in $O(n^4)$ time, or in time $O((n^2+m)\log n)$.

Next we relax the "general position" assumption. Two types of degeneracies can occur in the arrangement $\mathcal{A}(\mathcal{Q})$: (i) more than two arcs are incident to a vertex of the arrangement, and (ii) two or more arcs overlap in a subarc (not just in a point or two). We now show

that we can compute the arrangement and find all casting directions in asymptotically the same time as in the non-degenerate case.

Consider first a degeneracy of type (i), where a set Q_v of more than two arcs meet at a single vertex v. How to carry out the line sweep efficiently in this case is described in detail in [19, Chapter 2]. It remains to handle update of the counters at v. If every pair of arcs in Q_v cross each other transversally at v, then v requires no special treatment: if $\vec{d}_v$ is a valid casting direction then at least one of the edges of the arrangement incident to v also represents such directions, and the validity of the direction $\vec{d}_v$ will follow from its being an endpoint of that edge.

The case that requires caution is when in a small neighborhood of v, the vertex v is the only point representing a valid casting direction. This can happen when at least one of the arcs incident to v has an endpoint at v, or when two or more arcs overlap at and near v. The effect of overlap is explained below. In any case what is needed is careful counting at the vertex v, which can be carried out in time proportional to the number of incident edges, and hence can be charged to these edges. Clearly no edge is charged more than twice in this manner.

To handle degeneracies of type (ii) we carry out the following preprocessing step aiming to identify all overlaps among curves in $\mathcal{Q}$. By working on the projection plane as mentioned above, we can assign a slope to each great circle supporting an arc in $\mathcal{Q}$: the slope of the line it projects onto. We maintain the arcs sorted by slope. Let Q_s be the set of all arcs with the same slope s (hence potentially overlapping). All the arcs in Q_s lie on the great circle G_s. The endpoints of arcs in Q_s partition G_s into maximal intervals such that each interval is covered by the same set of arcs. These intervals constitute the new curves that we give as input to the algorithm that constructs the arrangement. By doing a 1-dimensional sweep on G_s (with the endpoints of all arcs in Q_s in cyclic order as the sweep events) we can decide for each of the new curves how the counters change as we cross the curve. We repeat this for every slope, and obtain a new set of curves that are then input to the algorithm for constructing the arrangement. It may also be the case that there are no endpoints of arcs, when all the curves in a set Q_s are great circles—these are simply unified into one curve with the appropriate counter update information.

Of special interest are arcs where the counters are zero on the arc but grow when moving out of the arc (that is if we cross such arc transversally, then locally the counters are positive just before and just after the crossing, but they are zero when we are on the arc). These arcs are interesting because if two of them meet transversally at a vertex v, this vertex v is potentially a secluded (singular) valid casting direction.

Let N denote the number of curves in $\mathcal{Q}$ and let m denote as before the complexity of the arrangement $\mathcal{A}(\mathcal{Q})$. Sorting the arcs by slope takes $O(N \log N)$ time and this is also the overall time to carry out all the 1-dimensional sweeps over the Q_s's. Other than that the algorithm is carried out as before. Since $N = O(n^2)$, the asymptotic running time of the algorithm in the general case remains $O((n^2 + m)\log n)$. We summarize with the

following:

Theorem 4 *Let $\mathcal{P}$ be a simple polyhedron with n vertices. All directions in which there is a good cast can be computed in $O(n^4)$ time. Alternatively, all the directions in which there is a good cast can be computed in $O((n^2+m)\log n)$ time, where m is the combinatorial complexity of the arrangement $\mathcal{A}(\mathcal{Q})$, or in expected time $O(n^2 \log n + m)$.*

We conclude this section by presenting a lower bound construction of polyhedra that have as many as $\Omega(n^4)$ distinct cast directions. This implies that our algorithm is optimal in the worst case. The key idea behind the construction of such polyhedra is to force the $\Omega(n^2)$ great circle arcs (formed by vertices crossing edges) to interact such that there are $\Omega(n^4)$ cells in the resulting arrangement.

Theorem 5 *There exist polyhedra for which there are $\Omega(n^4)$ distinct directions in which there is a good cast.*

Proof: Figure 3.10 shows a polyhedron with two horizontal "legs" and a row of small (resp. large) "teeth" positioned along the upper leg (resp. lower leg). We refer to this polyhedron as a *comb*. The schematic diagram on the left in Figure 3.11 gives the top view showing the interaction of the large and small teeth in a single comb. In the top view, if we move from left to right, one large tooth will appear in each of the gaps among the small teeth before an adjacent large tooth appears in any gap among the small teeth. Therefore, if there are b small teeth and c large teeth, then there are $\Omega(bc)$ distinct and good parting directions for a comb. For one comb, each of these distinct and good parting directions lies in a distinct cell in the arrangement, $\mathcal{A}(\mathcal{Q})$.

The key to increasing the number of distinct and good parting directions is to combine two combs. The schematic diagram on the right in Figure 3.11 shows the top view of a composite object consisting of two combs: the lower leg of the left comb and the upper leg of the right comb are at the same level, and the two shaded boxes represent the projection of the two rows of small teeth. Let each comb have a row of $n/4$ small and large teeth. Therefore, there is a total of n teeth in the composite object.

The first comb, in the composite object, decomposes the sphere of directions into $\Omega(n^2)$ cells. For each of these cells, the second comb decomposes that cell into $\Omega(n^2)$ cells. Therefore, the number of distinct and good parting directions for the composite object can be as large as $\Omega(n^4)$. □

3.4 Experimental results

We have implemented a simplified version of the algorithm of Theorem 3 (instead of a plane sweep, we simply test all pairs of shadow edges for crossings). Given an object

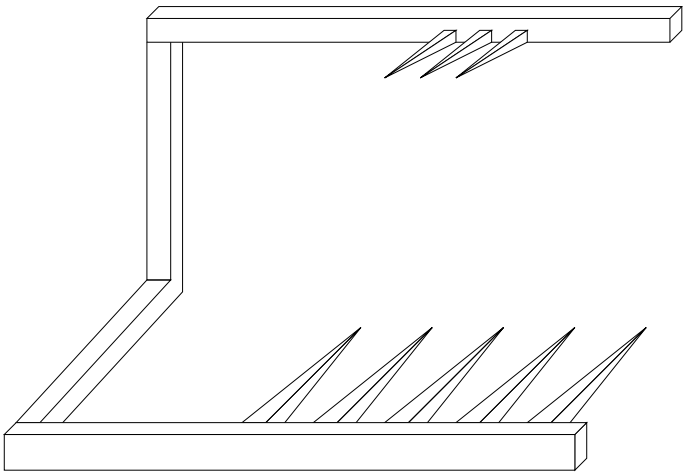


Figure 3.10: A component of the polyhedron having $\Omega(n^4)$ distinct cast directions

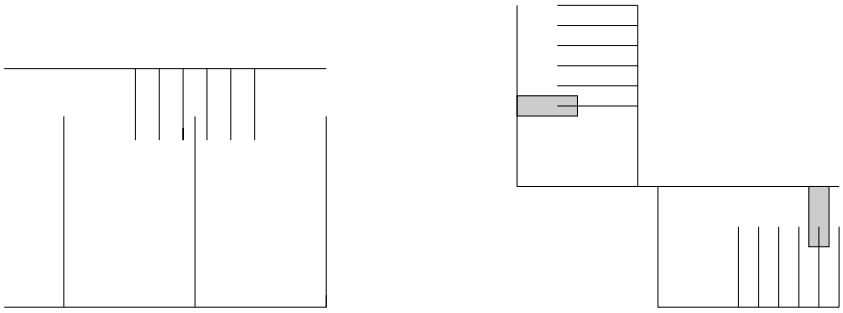


Figure 3.11: A top view of the lower bound construction

to be cast, we test a random set of directions, as well as heuristically chosen directions (currently the directions of all edges). This is simple to implement and seems to work fine in practice. Figure 3.12 shows a heart-shaped object. The sphere represents the sphere of directions. A black stipple is plotted on the sphere for every direction in which the object has been found castable. Two directions have been specially marked, one of these is an edge direction, the other one a randomly chosen direction. The heart has 75 edges and 7 edge directions were found to be feasible. Of the $32,000$ randomly chosen directions, 803 were found to be feasible. The black line on the heart shows the silhouette for one of

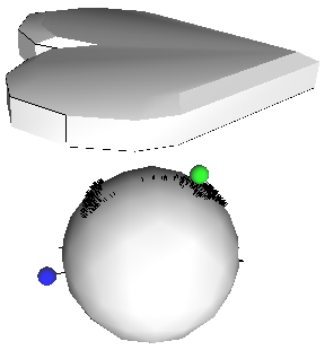


Figure 3.12: A heart-shaped object and the sphere of directions depicting the different casting directions tested

the two marked directions (the randomly chosen one).

The rook from a chess game in Figure 3.13 had 264 edges and the directions of 8 of them are feasible. Of the tested $32,000$ random directions, 576 are feasible. Silhouettes for the two marked directions are shown on the object.

We have also tested the implementation on many CAD models. In the implementation, we used the commercial software ACIS,[2] a 3-dimensional modeling toolkit and library, which supports solid modeling and surface modeling.

Figure 3.14 shows a convex polyhedron, which is castable in any direction. In the right figure, silhouettes for the object with respect to the given direction are shown. Note that the given direction is not feasible in the sand casting model.

The algorithm works for objects having tunnels. The torus in Figure 3.15 has a tunnel, and silhouettes of two closed curves were found with respect to the given direction.

Given a direction, if there is a reflex edge or a pair of edges that cross in the projection, the program not only stops, but provides useful information to the designer on the castability: Figure 3.16 shows a bracket. The bracket is not castable in the given direction, and a short black segment shown in the right figure is a reflex edge in the direction. Figure 3.17 shows an object whose two edges, colored black, cross in the projection into the given direction.

Figure 3.18 shows a bolt. Instead of sampling directions randomly, we chose directions parallel to faces of the object. Of the 64 chosen directions, 38 of lighter shade are feasible.

Clearly, further experimentation is necessary to improve the heuristics. Although the

[2] ACIS is a trademark of Spatial Technology Inc.

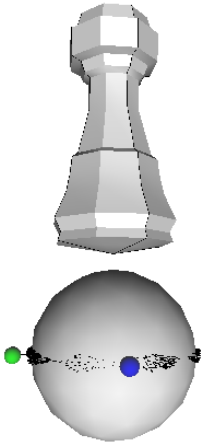


Figure 3.13: A rook and the sphere of directions depicting the different casting directions tested

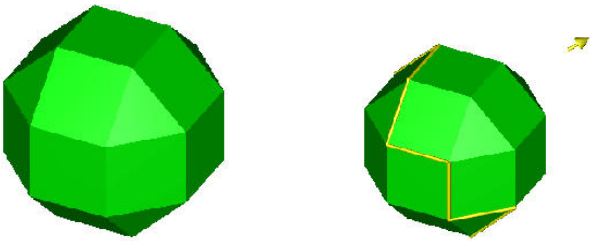


Figure 3.14: For a convex polyhedron, in any direction it is castable.

program takes only a few seconds on these parts, it would have to be faster to allow on-line warnings inside a CAD system (we imagine a system that automatically warns the designer as soon as the object becomes uncastable).

A natural extension would be to test directions that are parallel to a pair of facet planes. Obviously we could also test all $O(n^4)$ vertices of the arrangement on the sphere of directions (see the previous section), but so far it seems that this would make the program slower without helping much in practice.

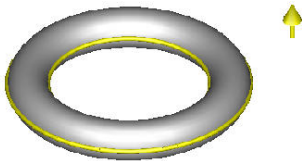


Figure 3.15: Torus: the algorithm works for objects having tunnels.

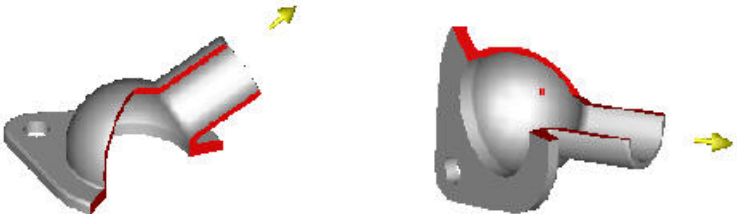


Figure 3.16: A reflex edge inside the bracket for the given direction.

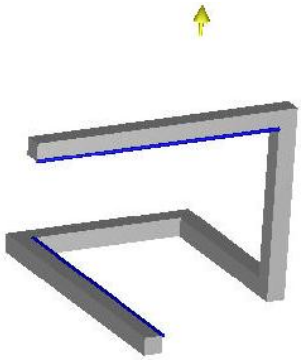


Figure 3.17: A pair of edges(dark) cross in the projection into the given direction

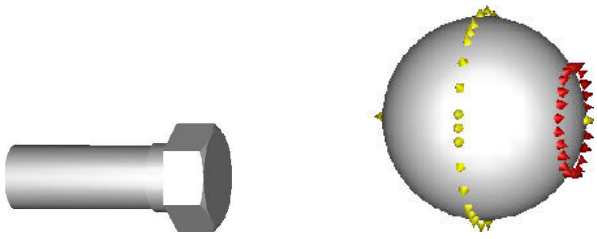


Figure 3.18: Directions parallel to faces of the object: lighter(castable)/darker(not castable)

CHAPTER 4

Directional uncertainty

4.1 Introduction

The casting algorithms that have been proposed so far assume perfect control of the casting machinery. When a cast part is removed, it is required that the part moves exactly in the specified direction. In practice, however, this will rarely be the case. As in all applications of robotics, we have to deal with imperfect control of the machinery, and a certain level of uncertainty in its movements. When a facet of the object or of a cast part is almost parallel to the direction in which the cast parts are being moved, then the two touching surfaces may damage each other when the cast (mould) is being opened. This can make the resulting object worthless, or it may wear away the surface of the cast so that it cannot be reused as often as desirable.

In Figure 4.1 (a), the cast can be opened by moving the two parts in direction $\vec{d}$ and $-\vec{d}$. If, however, due to imperfect control, the upper part is translated in direction $\vec{d'}$, it will destroy the object. The cast parts in (b) are redesigned so that both cast parts can be translated without damage in the presence of some uncertainty.

In this chapter, we consider directional uncertainty in the casting process: given a 3-dimensional polyhedral object, is there a polyhedral cast such that its two parts can be removed in opposite directions *with uncertainty* α without damage to the object or the cast parts? We call such an object *castable with uncertainty* α.

Directional uncertainty has been well studied by researchers in motion planning and robotics in general. A motion planning model with directional uncertainty was perhaps first proposed by Lozano-Pérez, Mason and Taylor [36]. An extensive treatment of motion planning with directional uncertainty is given in the book by Latombe [35].

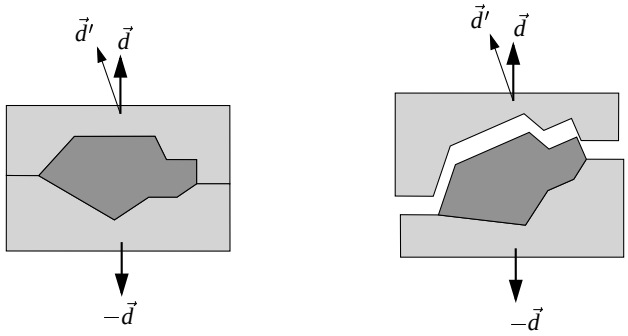


Figure 4.1: (a) Insufficient angle: the upper part of the cast is stuck, (b) A new removal direction

We generalize the characterization of castable polyhedra in Chapter 3 to incorporate uncertainty in the directions in which the cast parts are removed. A formal definition of our model is given in Section 4.2. It turns out that one of the main difficulties is to guarantee that the two cast parts are *polyhedral*—while this is trivial in the exact case, it requires approximation of a curved surface in our model with uncertainty. We give an algorithm that verifies whether a polyhedral object of arbitrary genus is castable for a given direction of cast part removal and given uncertainty $\alpha > 0$. The running time of the algorithm is $O(n \log n)$, where n is the number of vertices of the input polyhedron. If the object is castable, the algorithm also computes two polyhedral cast parts with $O(n)$ vertices in total.

We then consider the case where the removal direction is not specified in advance. We give an algorithm that finds all possible removal directions in which the polyhedral object is castable with uncertainty α in time $O(n^2 \log n/\alpha^2)$.

4.2 Preliminaries

A polyhedron $\mathcal{P}$ is *monotone* in direction $\vec{d}$ if every line with direction $\vec{d}$ intersects the interior of $\mathcal{P}$ in at most one connected component. We say that $\mathcal{P}$ is α-monotone in direction $\vec{d}$ for an angle α with $0 \leq \alpha < \pi/2$ if $\mathcal{P}$ is monotone in direction $\vec{d}'$ for all directions $\vec{d}'$ with $\angle(\vec{d}, \vec{d}') \leq \alpha$,

We say that a facet f of a polyhedron $\mathcal{P}$ is α-*steep* in direction $\vec{d}$ if the angle β between a normal of f and $\vec{d}$ lies in the range $\pi/2 - \alpha \leq \beta \leq \pi/2 + \alpha$. A polyhedron $\mathcal{P}$ is called α-*safe* in direction $\vec{d}$ if none of its facets is α-steep for that direction.

We call a polyhedral terrain α-*safe* if the normal vector of the surface makes an angle of

at most $\pi/2 - \alpha$ with the vertical direction wherever it is defined.

A *cast $\mathcal{C}$ with opening direction $\vec{d}$* for a polyhedron $\mathcal{P}$ is a pair $(\mathcal{C}_r, \mathcal{C}_b)$ of two polyhedra $\mathcal{C}_r$ and $\mathcal{C}_b$, such that the interiors of $\mathcal{C}_r$, $\mathcal{C}_b$, and $\mathcal{P}$ are pairwise disjoint and the union $\mathcal{C}_r \cup \mathcal{P} \cup \mathcal{C}_b$ is a rectangular box with an edge parallel to $\vec{d}$ that completely contains $\mathcal{P}$ in its interior. We call $\mathcal{C}_r$ and $\mathcal{C}_b$ the *red cast part* and the *blue cast part* of $\mathcal{C}$.

A cast $\mathcal{C}$ with opening direction $\vec{d}$ is α-feasible, if for each pair of directions $(\vec{d}_r, \vec{d}_b)$ with $\angle(\vec{d}, \vec{d}_r) \leq \alpha$ and $\angle(-\vec{d}, \vec{d}_b) \leq \alpha$, the red cast part $\mathcal{C}_r$ can be translated to infinity in direction $\vec{d}_r$ without colliding with $\mathcal{P}$ or $\mathcal{C}_b$, and the blue cast part $\mathcal{C}_b$ can be translated to infinity in direction $\vec{d}_b$ without colliding with $\mathcal{P}$. Note that the order of removing the cast parts is actually irrelevant.

A polyhedron $\mathcal{P}$ is *α-castable in direction $\vec{d}$* if an α-feasible cast with opening direction $\vec{d}$ exists. For the special case $\alpha = 0$, we say that $\mathcal{P}$ is *castable* in direction $\vec{d}$.

The following simple lemma characterizes polyhedra castable in direction $\vec{d}$ [7].

Lemma 6 *A polyhedron $\mathcal{P}$ is castable in direction $\vec{d}$ if and only if it is monotone in direction $\vec{d}$.*

The main result of the present chapter is a generalization of this result to α-castability. We state the result here—it will take us a few more pages to prove it.

Theorem 6 *A polyhedron $\mathcal{P}$ is α-castable in a direction $\vec{d}$ if and only if $\mathcal{P}$ is α-monotone and α-safe in direction $\vec{d}$.*

The following lemma proves the necessity of the condition.

Lemma 7 *If a polyhedron $\mathcal{P}$ is α-castable in direction $\vec{d}$, then $\mathcal{P}$ is α-monotone and α-safe in direction $\vec{d}$.*

Proof: Assume that $\mathcal{P}$ is not α-safe, so a facet f is α-steep with respect to $\vec{d}$. A point p in the interior of f can be neither on the boundary of $\mathcal{C}_r$ nor on the boundary of $\mathcal{C}_b$, and so $\mathcal{P}$ is not α-castable in direction $\vec{d}$.

On the other hand, if $\mathcal{P}$ is α-castable in direction $\vec{d}$, it is castable in any direction $\vec{d}'$ with $\angle(\vec{d}, \vec{d}') \leq \alpha$. By Lemma 6, it follows that $\mathcal{P}$ is monotone in direction $\vec{d}'$. It follows that $\mathcal{P}$ is α-monotone. □

4.3 Finding a cast

It remains to prove the sufficiency of the condition in Theorem 6. We do so by showing how to construct an α-feasible cast for any α-monotone and α-safe polyhedron. To

simplify the presentation, we will assume, without loss of generality, that $\vec{d}$ is the upward vertical direction (the positive z-direction). We say that $\mathcal{P}$ is α-castable if it is α-castable in the vertical direction.

A facet of $\mathcal{P}$ is called an *up-facet* if its outward normal points upwards, and a *down-facet* if its outward normal points downwards. Assuming $\mathcal{P}$ is α-safe, there are no vertical facets, and so each facet is either an up-facet or a down-facet. Clearly an up-facet of $\mathcal{P}$ must be a facet of the red cast part $\mathcal{C}_r$, while a down-facet of $\mathcal{P}$ must be a facet of the blue cast part $\mathcal{C}_b$. The difficulty is finding the separating surface between $\mathcal{C}_r$ and $\mathcal{C}_b$ "elsewhere."

Assume that $\mathcal{P}$ is α-castable and that $(\mathcal{C}_r, \mathcal{C}_b)$ is an α-feasible cast for $\mathcal{P}$. Again we denote by B the axis-parallel box that forms the outside of the cast. We define the *blue parting surface* S_b as the common boundary of $\mathcal{C}_b$ and $\mathcal{C}_r \cup \mathcal{P}$, and the *red parting surface* S_r as the common boundary of $\mathcal{C}_r$ and $\mathcal{C}_b \cup \mathcal{P}$. Any upwards directed vertical line ℓ must intersect $\mathcal{C}_b$, $\mathcal{P}$ and $\mathcal{C}_r$ in this order, each in a single connected component that can be empty. It follows that both S_b and S_r are polyhedral terrains. The two terrains coincide except where they bound the polyhedron $\mathcal{P}$. If we let $\mathcal{S} := S_b \cap S_r$, define $\mathcal{S}_u$ to be the union of all up-facets, and $\mathcal{S}_d$ to be the union of all down-facets, we can write $S_r = \mathcal{S} \cup \mathcal{S}_u$ and $S_b = \mathcal{S} \cup \mathcal{S}_d$. The boundary of $\mathcal{S}$ is the set of silhouette edges of $\mathcal{P}$ (an edge is a silhouette edge if it separates an up-facet from a down-facet).

Constructing a cast therefore reduces to the construction of the terrain $\mathcal{S}$. In Chapter 3, we considered the special case $\alpha = 0$, and gave a triangulation method for constructing $\mathcal{S}$ as follows: Let h be a horizontal plane cutting the box B in two roughly equal halves. Let R be the rectangle $h \cap B$. We project $\mathcal{P}$ onto h and obtain a polygon $\overline{\mathcal{P}}$, possibly with holes. Let $\mathcal{T}$ be a triangulation of $R \setminus \overline{\mathcal{P}}$. Every triangle in $\mathcal{T}$ is "lifted" into 3-dimensional space by replacing each vertex $\overline{v}$ of $\overline{\mathcal{P}}$ by its original vertex v of $\mathcal{P}$. The resulting 3-dimensional surface is the desired terrain $\mathcal{S}$ separating the red and blue cast parts. (The description in Chapter 3 is more complicated as it handles vertical facets.)

Unfortunately, this construction does not necessarily produce an α-feasible cast, even when the polyhedron is α-castable. Figure 4.2 illustrates this possibility. $\overline{\mathcal{P}}$ is the projection of a polyhedron $\mathcal{P}$ that is α-monotone and α-safe. The z-coordinates of vertices a and b are identical (and so the segment ab is horizontal). The z-coordinate of c is chosen such that both ac and bc make an angle of α with the vertical direction. Any triangulation of $R \setminus \overline{\mathcal{P}}$ contains the triangle abc. This implies that the midpoint p of ab lies on $\mathcal{S}$, and therefore on the boundary of the red cast part. However, translating p upwards with uncertainty α may cause it to collide with the polyhedron at c, and so the cast is not α-feasible.

The problem with this approach is that even if the polyhedron is α-monotone and α-safe, the constructed terrain $\mathcal{S}$ is not: the triangle abc is in fact α-steep. We now prove that it suffices to make sure this does not happen.

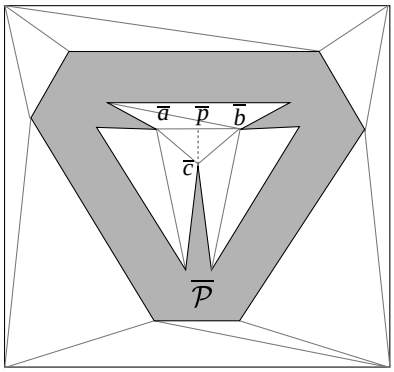


Figure 4.2: The triangulation method fails: the line segment pc is too steep.

Lemma 8 *Let B be an axis-parallel box, and let $\mathcal{S}$ be an α-safe terrain separating the top and bottom facets of B. Let $\mathcal{C}$ be the part of B above $\mathcal{S}$, and let $\mathcal{C}' := B \setminus \mathcal{C}$. Let $\vec{d}$ be the upward vertical direction, and let $\vec{d}'$ be such that $\angle(\vec{d}, \vec{d}') \leq \alpha$. Then $\mathcal{C}$ can be translated to infinity in direction $\vec{d}'$ without colliding with $\mathcal{C}'$.*

Proof: Assume the claim was false, and consider a point $p \in \mathcal{C}$ that when translated in direction $\vec{d}'$ collides with a point $q \in \mathcal{C}'$. The line segment pq lies completely inside B, and so its vertical projection onto $\mathcal{S}$ is a path π. Since p lies above one end-point of π, q lies below the other end-point, and the slope of pq is at least $\pi/2 - \alpha$, there must be a segment on π where the slope is at least $\pi/2 - \alpha$. This is a contradiction to the assumption that $\mathcal{S}$ is α-safe. □

Lemma 9 *Let $\mathcal{P}$ be an α-safe polyhedron, B an axis-parallel box enclosing $\mathcal{P}$ and let $\mathcal{S}$ be an α-safe polyhedral terrain bounded by the silhouette edges of $\mathcal{P}$. Then the cast defined by the parting surfaces $S_r := \mathcal{S} \cup \mathcal{S}_u$ and $S_b := \mathcal{S} \cup \mathcal{S}_d$ is α-feasible.*

Proof: Since $\mathcal{P}$ is α-safe, both $\mathcal{S}_u$ and $\mathcal{S}_d$ are α-safe terrains. Since $\mathcal{S}$ is α-safe, both S_r and S_b are therefore α-safe. Lemma 8 now implies that the cast is α-feasible. □

We will now show how to construct a terrain $\mathcal{S}$ as in Lemma 9 by forming the lower envelope of a set of cones. Given a point p on an up-facet of $\mathcal{P}$, the α-*cone* $\mathcal{D}(p)$ of p is the solid vertical upwards oriented cone of angle α with apex p. Formally, if p' is a point vertically above p, then $\mathcal{D}(p) := \{x \mid \angle(xpp') \leq \alpha\}$. Let now $\mathcal{D}_1$ be the union of $\mathcal{D}(p)$ over all points $p \in \mathcal{S}_u$, and let $\mathcal{E}_1$ be the lower envelope of $\mathcal{D}_1$. Clearly, $\mathcal{E}_1$ contains $\mathcal{S}_u$, and so $\mathcal{S} := \mathcal{E}_1 \setminus \mathcal{S}_u$ is bounded by the silhouette edges of $\mathcal{P}$. Since $\mathcal{E}_1$ consists of patches

of α-cones, it is clearly α-safe. It follows that $\mathcal{S}$ fulfills the requirements of Lemma 9, except that it is not a polyhedral terrain.

We will see below that we can easily "approximate" $\mathcal{S}$ by a polyhedral, α-safe terrain $\mathcal{S}'$ that contains all the linear edges of $\mathcal{S}$ and lies below (or coincides with) $\mathcal{S}$ everywhere. (The reader might also rightfully ask why a cast has to be polyhedral—perhaps a cast bounded by the conic patches resulting from our construction might work better in practice than the polyhedral version we will construct below.)

The construction of $\mathcal{S}$ above appears to require taking the union of an infinite family of cones. We now give an alternative definition of $\mathcal{S}$ as the lower envelope of h objects, where h is the number of silhouette edges of $\mathcal{P}$.

In fact, let pq be a silhouette edge of $\mathcal{P}$. The α-region $\mathcal{D}(pq)$ of pq is the convex hull of $\mathcal{D}(p) \cup \mathcal{D}(q)$. The lower envelope of $\mathcal{D}(pq)$ consists of three components: two conic surfaces supported by the α-cones $\mathcal{D}(p)$ and $\mathcal{D}(q)$, and a connecting area consisting of two planar facets.

Let now $\mathcal{D}_2$ be the union of $\mathcal{D}(pq)$, over all silhouette edges pq, and let $\mathcal{E} = \mathcal{E}_2$ be the lower envelope of $\mathcal{D}_2$. It is easy to see that $\mathcal{E}_1$ is in fact the lower envelope of $\mathcal{S}_u$ and $\mathcal{E}_2$, and so $\mathcal{E}_1$ and $\mathcal{E}_2$ coincide "outside" of $\mathcal{P}$. Thus, if we define $\mathcal{S}$ to be the part of $\mathcal{E} = \mathcal{E}_2$ not lying above $\mathcal{S}_u$, we define the same terrain $\mathcal{S}$ as above.

The lower envelope $\mathcal{E}$ consists of $O(h)$ *faces*, which are either planar, or supported by a single α-cone $\mathcal{D}(x)$ for a vertex x of $\mathcal{P}$. An edge of $\mathcal{E}$ is either a silhouette edge of $\mathcal{P}$, a straight edge separating a conic patch supported by an α-cone $\mathcal{D}(x)$ from an adjacent planar patch supported by an α-region $\mathcal{D}(xy)$, or is an arc supported by the intersection curve of two α-cones, an α-cone and a plane, or two planes. Such arcs are either straight segments, arcs of parabolas, or arcs of hyperbolas. In all cases, they are contained in a plane. Figure 4.3 shows the two types of conic sections arising.

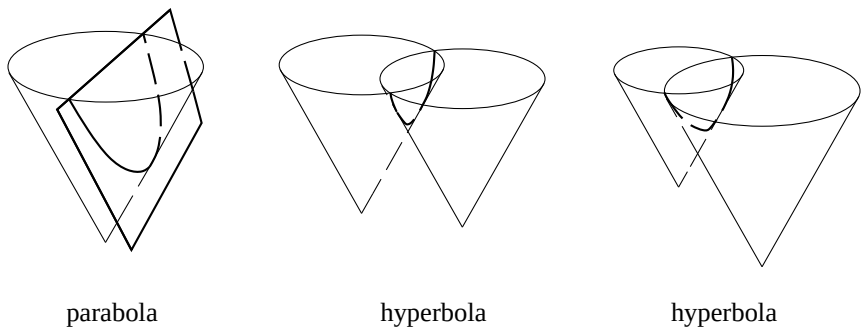


Figure 4.3: Types of conic sections: parabola and hyperbola

We can represent $\mathcal{E}$ by its projection on the *xy*-plane. The projection is in fact a planar subdivision, whose faces are supported by a single plane or α-cone. If we annotate each face with the vertex or silhouette edge of $\mathcal{P}$ whose α-cone or α-region supports it, the resulting map is a complete representation of $\mathcal{E}$.

In general, the lower envelope of m well-behaved, constant-complexity objects can have complexity $\Theta(m^2)$ [48]. We will show in the following that our planar subdivision has in fact *linear* complexity. Roughly speaking, we interpret the planar map as a kind of Voronoi diagram. Our sites are the projections of silhouette edges onto the *xy*-plane, additively weighted by the "height" of the edge above the *xy*-plane. (This is, indeed, a strange notion of "weight," as it is not constant for a given site. The concerned reader is asked to wait for the formal definition below.) This diagram does not appear to have been studied before, but it does fit into Klein's framework of *abstract Voronoi diagrams* [32], and his results on complexity and computation apply.

Consider a silhouette edge e of $\mathcal{P}$. Let $\overline{e}$ be the projection of e on the *xy*-plane. For a point $\overline{p} \in \overline{e}$, let p_z be the *z*-coordinate of the point $p = (p_x, p_y, p_z) \in e$ whose projection on the *xy*-plane is $\overline{p}$, and let $w(\overline{p})$ be $p_z \tan\alpha$. We can now define a distance measure in the plane as follows: For $x \in \mathbb{R}^2$ and $\overline{p} \in \overline{e}$, we define

$$d(x, \overline{p}) := |x\overline{p}| + w(\overline{p}) = |x\overline{p}| + p_z \tan\alpha.$$

The distance of a point x to a segment $\overline{e}$ is then

$$d(x, \overline{e}) := \min_{\overline{p} \in \overline{e}} d(x, \overline{p}).$$

Lemma 10 *The vertical projection of the lower envelope $\mathcal{E}$ coincides with the Voronoi diagram of the projected silhouette edges and vertices under the distance function defined above.*

Proof: Let x be a point in the plane, and let e be a silhouette edge of $\mathcal{P}$. Let x^* be the point where the vertical line through x intersects the boundary of the α-region $\mathcal{D}(e)$. We observe that $d(x, \overline{e}) = |xx^*| \tan\alpha$. The lemma follows. □

In the following lemma, we show some properties of the Voronoi diagram.

Lemma 11 *Let $\mathcal{P}$ be an α-safe and α-monotone polyhedron. Consider the Voronoi diagram defined by the projections of a subset G' of silhouette edges of $\mathcal{P}$ with the distance function above. It has the following properties:*

- *A projected silhouette edge $\overline{e}$ lies in its own Voronoi cell.*
- *Given a point x in the Voronoi cell of $\overline{e}$. Let $y \in \overline{e}$ be the point on $\overline{e}$ minimizing the distance from x. Then the segment xy is contained in the Voronoi cell of $\overline{e}$.*

- *Each Voronoi cell is simply connected.*
- *The Voronoi diagram is an abstract Voronoi diagram as defined by Klein et al. [33].*

Proof: Let G' be a non-empty subset of silhouette edges, and let $\mathcal{E}'$ be the lower envelope of the α-regions of the silhouette edges in G'.

(i) The claim is identical to stating that the silhouette edge e appears on the lower envelope $\mathcal{E}'$. If it didn't, a point $p \in e$ would have to lie inside the α-region $\mathcal{D}(e')$ of some other silhouette edge e', in contradiction to the assumption that $\mathcal{P}$ is α-monotone.

(ii) Assume there is a point $z \in xy$ such that the nearest site point to z is $t \neq y$. Then

$$\begin{aligned} d(x,t) &= |xt| + w(t) \leq |xz| + |zt| + w(t) = |xz| + d(z,t) \\ &< |xz| + d(z,y) = |xz| + |zy| + w(y) = |xy| + w(y) = d(x,y), \end{aligned}$$

in contradiction to the definition of y. So the nearest point on a site is y, for all points on xy, and the segment xy is contained in the Voronoi cell of $\overline{e}$.

(iii) Follows from (i) and (ii).

(iv) The abstract Voronoi diagram framework by Klein et al. assumes a set of (abstract) objects, each pair of which defines a bisector partitioning the plane into two unbounded regions. The system of bisectors has to adhere to a set of four axioms. It is straightforward to verify that the bisectors defined by pairs of silhouette edges do fulfill these axioms, using (i)-(iii) and elementary calculations. □

Figure 4.4 shows the bisector of two projected silhouette edges $\overline{e}$ and $\overline{e'}$. Note the drop-shaped curves surrounding each edge: these are curves of equal distance from the segment.

Lemma 12 *Let $\mathcal{P}$ be an α-monotone and α-safe polyhedron with n vertices, and let $\mathcal{E}$ be the lower envelope of the α-regions of its silhouette edges. Then $\mathcal{E}$ has complexity $O(n)$ and can be computed in time $O(n \log n)$.*

Proof: From Lemma 10 and Lemma 11 (i)-(iii), we can conclude that $\mathcal{E}$ has linear complexity.

We can identify the h silhouette edges of $\mathcal{P}$ in $O(n)$ time by inspecting the normals of all facets. By Lemma 11 (iv), the projection of $\mathcal{E}$ onto the xy-plane can be computed in time $O(h \log h)$ by the randomized incremental algorithm of Klein et al. [33]. Each face of the Voronoi diagram carries information about the site creating it, and so we can construct the envelope $\mathcal{E}$ in linear time $O(h)$. □

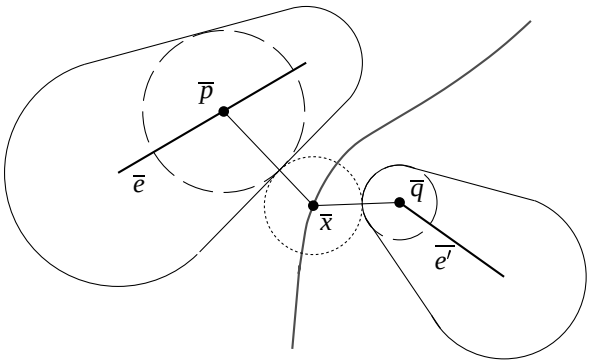


Figure 4.4: $d(\overline{e},\overline{x}) = d(\overline{e'},\overline{x})$

We have now seen how to compute an α-safe terrain $\mathcal{E}$ bounded by the silhouette edges of $\mathcal{P}$ in time $O(n \log n)$. All that remains to be done to fulfill the assumptions of Lemma 9 is to turn $\mathcal{E}$ into a *polyhedral* terrain. We proceed as follows.

The edges of $\mathcal{E}$ consist a constant number of segments of two types: straight line segments and conic arcs. Let $\delta = v_1 v_2$ be such a conic arc, with endpoints v_1 and v_2. Its projection $\overline{\delta}$ separates two cells of the Voronoi diagram, say of $\overline{e}$ and $\overline{e'}$.

We conceptually add four straight line segments to the graph of the Voronoi diagram by connecting both $\overline{v_1}$ and $\overline{v_2}$ to the nearest point on both $\overline{e}$ and $\overline{e'}$. We do this for all conic arcs of $\mathcal{E}$, adding a linear number of "spokes" to the Voronoi diagram graph. The spokes do not intersect, and so we have increased the complexity of the diagram by a constant factor only. As a result, any conic arc $\overline{\delta}$, is now incident to two constant-complexity faces in the diagram. There are two cases, depicted in Figure 4.5 (a), depending on whether the spokes meet on one or two sides. Without loss of generality, we can assume that the spokes always meet on $\overline{e'}$.

As we have seen before, the conic arc δ is contained in a plane Γ. We now choose a line ℓ in Γ tangent to δ on its convex side, such that its projection $\overline{\ell}$ separates $\overline{\delta}$ from $\overline{e}$. (If Γ is a vertical plane, then $\overline{\delta}$ is a straight segment, and $\overline{\ell}$ contains $\overline{\delta}$.) Let furthermore ℓ_1 and ℓ_2 be the lines in Γ tangent to δ in v_1 and v_2. Let $x := \ell \cap \ell_1$ and $y := \ell \cap \ell_2$.

We now construct a new terrain $\mathcal{E}'$ by replacing the conic arc δ with the polygonal chain $v_1 x y v_2$, and replacing the conic surface patches supported by $\mathcal{D}(e')$ and $\mathcal{D}(e)$ each by three triangles $e' v_1 x$, $e' x y$, $e' y v_2$ (and analogously for e if the spokes meet on both sides). Figure 4.5 (b) shows the projection of the new terrain $\mathcal{E}'$.

We can perform this operation for all conic arcs of $\mathcal{E}$ simultanously, resulting in a polyhedral terrain $\mathcal{E}'$. Note that the triangles lie on planes that are tangent to α-cones $\mathcal{D}(e)$

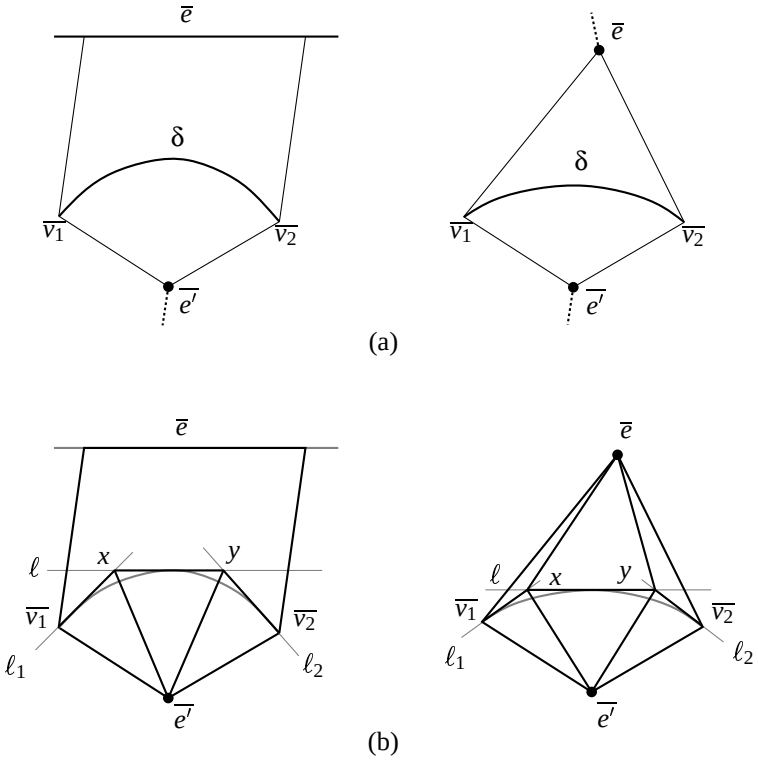


Figure 4.5: Approximation of curved surfaces. (a) adding spokes to the diagram: bisector of a vertex and an edge (left) and bisector of two vertices (right). (b) in the new terrain $\mathcal{E}'$, conic facets have been replaced by triangles.

or $\mathcal{D}(e')$, and so they are not α-steep. This implies that $\mathcal{E}'$ is α-safe. By Lemma 9, the terrain $\mathcal{E}'$ defines an α-feasible cast, and we have the following result.

Lemma 13 *If a polyhedron $\mathcal{P}$ is α-monotone and α-safe in direction $\vec{d}$, then $\mathcal{P}$ is α-castable in direction $\vec{d}$.*

This concludes the proof of Theorem 6; the theorem follows immediately from Lemmas 7 and 13.

Our proof of Theorem 6 is constructive: Given an α-safe and α-monotone polyhedron, we can compute a feasible cast with uncertainty α in time $O(n \log n)$. The construction uses the randomized incremental algorithm by Klein et al. [33], as in Lemma 12.

This procedure does not yet allow us to *decide* whether a polyhedron is indeed α-castable, as we can only guarantee the correctness of the envelope construction if the envelope is

indeed an abstract Voronoi diagram. This is not necessarily the case if the polyhedron is not α-monotone. Fortunately, it is not difficult to add a test to each stage of the algorithm by Klein et al. that will detect if $\mathcal{P}$ is not α-monotone. This is based on the following lemma.

Lemma 14 *Let $\mathcal{P}$ be α-safe and monotone, and let G' be a non-empty subset of the silhouette edges of $\mathcal{P}$. G' is α-monotone if and only if each edge $e \in G'$ appears completely on the lower envelope of the α-regions of G'.*

Proof: The necessity of the condition was already proven in Lemma 11.

Assume that G' is not α-monotone. Then there are two silhouette edges e, e' and two points $p \in e$ and $q \in e'$ such that the slope of pq is greater than $2\pi - \alpha$. The point p, therefore, lies inside the α-region $\mathcal{D}(e')$, and so e does not appear completely on the lower envelope. □

We can now augment the algorithm by Klein et al. to achieve the following result.

Theorem 7 *Given a polyhedron $\mathcal{P}$ with n vertices and a direction $\vec{d}$, we can test the α-castability of $\mathcal{P}$ in $\vec{d}$ in time $O(n \log n)$. If it is α-castable, then we can construct an α-feasible cast in $O(n \log n)$ time. The resulting cast has $O(n)$ vertices.*

Proof: We first examine every facet of $\mathcal{P}$ and decide whether $\mathcal{P}$ is α-safe. If so, we test whether $\mathcal{P}$ is monotone in direction $\vec{d}$, for instance using the algorithm in Chapter 3. If either step fails, we report that $\mathcal{P}$ is not α-castable in direction $\vec{d}$.

Otherwise, we now use the algorithm by Klein et al. [33] to compute the Voronoi diagram of the projected silhouette edges G. The algorithm incrementally constructs the diagram, while adding the projecting silhouette edges one by one in random order. At each step, it maintains the Voronoi diagram $V(\overline{G'})$ and a so-called history graph $\mathcal{H}(\overline{G'})$ of the subset G' of edges inserted so far. When inserting a new silhouette edge $s \in \{G \setminus G'\}$, the algorithm first computes the set E_s of Voronoi edges that are intersected by the Voronoi region $V(s)$ in $V(\overline{G' \cup \{s\}})$. Then it constructs the updated diagram $V(\overline{G' \cup \{s\}})$ and the updated history graph $\mathcal{H}(\overline{G' \cup \{s\}})$ by using E_s. This can be done in $O(E_s)$ time [33].

We know that this procedure works correctly as long as the subset G' is α-monotone. We augment the algorithm such that it recognizes, as soon as a new silhouette edge $s \in G \setminus G'$ is added, whether $G' \cup \{s\}$ is no longer α-monotone. The test relies on Lemma 14. $G' \cup \{s\}$ is not α-monotone if and only if there is $s' \in G'$ such that either $s \cap \mathcal{D}(s') \neq \emptyset$ or $s' \cap \mathcal{D}(s) \neq \emptyset$. The silhouette edge s' must participate in the definition of the Voronoi edges of E_s, and so we can test this in $O(E_s)$ time. It now suffices to verify that E_s is indeed correctly computed by the algorithm even if $G' \cup \{s\}$ is not α-monotone. □

4.4 Computing feasible directions

We now describe an algorithm to solve the following problem: Given a polyhedron $\mathcal{P}$ and an angle α, decide whether there is a direction $\vec{d}$ such that $\mathcal{P}$ is α-castable in direction $\vec{d}$. In fact, we will solve the more general problem of finding all directions $\vec{d}$ for which $\mathcal{P}$ is α-castable.

We identify the set of directions with the set of points on the unit sphere $\mathcal{S}^2$ centered at the origin. A point p on $\mathcal{S}^2$ corresponds to the direction $\vec{d}_p$ from the origin o to p. Our goal is to identify the region of $\mathcal{S}^2$ corresponding to directions in which $\mathcal{P}$ is α-castable.

If we imagine the direction $\vec{d}$ changing continuously, there are directions where an up-facet may become a down-facet, or vice versa. The set of these directions forms a collection M of $O(n)$ great circles on $\mathcal{S}^2$. We note that $\mathcal{P}$ is α-safe in a direction $\vec{d}_p$ if and only if p has distance at least α to all great circles in M.

Let C be a cell of the great circle arrangement of M. If $\vec{d}$ varies inside C, the silhouette edges of $\mathcal{P}$ remain the same, but at certain directions the monotonicity of $\mathcal{P}$ changes. In fact, this happens when a line parallel to $\vec{d}$ through a silhouette vertex crosses a silhouette edge. The set of directions for which this occurs forms a collection N of $O(n^2)$ arcs of great circles. We note that $\mathcal{P}$ is α-monotone in direction $\vec{d}_p$ if and only if $\mathcal{P}$ is monotone in direction $\vec{d}_p$ and p has distance at least α to all the arcs in N.

Instead of computing the complete arrangement of $M \cup N$, we can work with a set S of $O(1/\alpha^2)$ sampling points on $\mathcal{S}^2$. The sampling points S are chosen such that any spherical disc of radius α on $\mathcal{S}^2$ contains a point of S.

For each $s \in S$, we first test whether $\mathcal{P}$ is monotone in direction $\vec{d}_s$ in time $O(n \log n)$, using the algorithm in Chapter 3. If it is, we construct the cell of the arrangement of M containing s by computing the intersection C_1 of n hemispheres in time $O(n \log n)$. We then compute the $O(n^2)$ arcs of great circles where the monotonicity of $\mathcal{P}$ changes within C_1, and compute the single cell C_2 containing s in their arrangement in time $O(n^2 \log n)$ using the randomized incremental construction algorithm by de Berg et al. [18]. By the observations above, if $p \in C_2$ then $\mathcal{P}$ is α-monotone and α-castable in direction $\vec{d}$ if and only if p has distance at least α to the boundary of C_2. We can compute this set of directions by taking the Minkowski-difference of C_2 and a disc of radius α.

It remains to argue that all feasible casting directions are found this way. Let $\vec{d}_p$ be a direction in which $\mathcal{P}$ is α-castable. The spherical disc with center p and radius α contains a point $s \in S$, and does not intersect any great circle arc in M or N. This implies that p and s are contained in the same cell of the arrangement of $M \cup N$. Furthermore, $\mathcal{P}$ must be monotone in direction $\vec{d}_s$. It follows that p will be found by our algorithm.

Finding the direction of maximum uncertainty. It is desirable that the parting terrain of a cast is as "flat" as possible. So while a relatively small uncertainty α may be given as a minimum requirement for manufacturing, we actually prefer to generate casts with

uncertainty as large as possible.

We can easily extend the algorithm described above to solve this problem. Again we are given an angle $\alpha > 0$ and wish to test whether $\mathcal{P}$ is α-castable. If the answer is positive, we now also want to determine the largest $\alpha^* > \alpha$ for which a direction $\vec{d}$ exists such that $\mathcal{P}$ is α^*-castable in direction $\vec{d}$.

We proceed as above: We generate a sampling set S such that any spherical disc of radius α contains a point of S. We then compute, for each $s \in S$, the cell C_2 containing s. The direction of largest uncertainty within C_2 is the center of the maximum inscribed (spherical) disc for C_2, which we compute in $O(n^2 \log n)$ time. The largest inscribed disc, over all cells computed, determines the largest uncertainty for which the object is still castable.

Theorem 8 *Let $\mathcal{P}$ be a polyhedron with n vertices, and $\alpha > 0$. All directions in which $\mathcal{P}$ is castable with uncertainty α can be computed in $O(n^2 \log n/\alpha^2)$ time. If such a direction exists, the largest $\alpha^* > \alpha$ for which $\mathcal{P}$ is castable with uncertainty α^* can be computed within the same time bound.*

A heuristic. If an approximative solution is sufficient, the following heuristic can be applied. It runs in time $O(n \log n)$ for constant α.

Let $\alpha' := (1-\varepsilon)\alpha$, for some approximation parameter $\varepsilon > 0$. We choose a set S of $O(1)$ sampling directions on $\mathcal{S}^2$, sufficiently dense such that for any spherical disc D of radius α there is a point $s \in S$ such that the disc of radius α' with center s is contained in D.

For each $s \in S$ we test whether $\mathcal{P}$ is α-castable using the algorithm of Section 4.3. If we are successful, we report $\mathcal{P}$ to be α-castable. If not, we test each direction $s \in S$ again, this time with uncertainty α'. If no feasible casting direction with uncertainty α' is found, we report that $\mathcal{P}$ is not castable with uncertainty α. This is true by the choice of S. If a feasible direction for uncertainty α' is found, we report a "maybe" answer: $\mathcal{P}$ is castable with uncertainty $(1-\varepsilon)\alpha$, and may or may not be castable with uncertainty α.

The same idea can be used to approximate the largest feasible uncertainty. We can, for instance, set $\alpha' := \alpha/2$, and keep doubling α until $\mathcal{P}$ is no longer α-castable.

CHAPTER 5

Skewed ejection direction

5.1 Introduction

In this chapter we study the case where the cast parts need not be removed in opposite directions. In most existing machinery, the retraction and ejection directions are identical, and previous work on this problem has assumed this restriction on casting. Existing technology for injection moulding, however, already has the flexibility to accommodate an ejection direction that is different from the retraction direction of the moving cast part. Exploiting this possibility allows to cast more parts, or to cast parts with simpler moulds, and is the subject of the present chapter. Figure 5.1 illustrates the process on a 2-dimensional example.

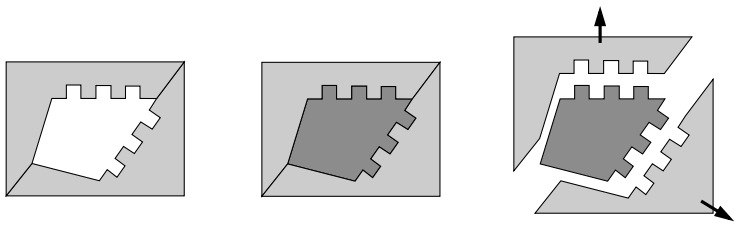


Figure 5.1: The casting process

To summarize, in our model of casting, the two cast parts are to be removed in two given directions and these directions need not be opposite. Contrary to the case of opposite

removal directions, the ordering of removal is important.

The cast parts should be removed from the object without destroying either cast parts or the object. This ensures that the given object can be mass produced by re-using the same cast parts. The casting process may fail in the removal of the cast parts: if the cast is not designed properly, then one or more of the cast parts may be stuck during the removal phase, as in Figure 5.2. The problem we address here concerns this aspect: Given a 3-dimensional object, is there a cast for it whose two parts can be removed after the liquid has solidified? An object for which this is the case is called *castable*.

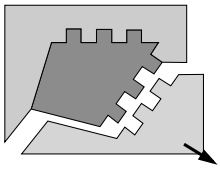


Figure 5.2: The top part of the cast is stuck.

Separating a cast in two arbitrary removal directions for 2 dimensions [45] and in some special cases for 3 dimensions have been studied before [13]. While in practice the two cast parts are removed in opposite directions, cores and inserts can be used to enlarge the class of objects manufacturable by casting [20, 41, 55]. Cores and inserts are appendages to the cast parts that are removed in arbitrary directions. Thus, our technique for handling two arbitrary removal directions may shed some light on the problem of incorporating cores and inserts.

In this chapter we give a complete characterization of castability, under the assumption that the cast has to consist of two parts that are to be removed in not necessarily opposite directions. Our characterization of castability applies to general objects $\mathcal{Q}$. This is important since many industrial parts are not polyhedral. We also give an algorithm to verify this condition for polyhedral objects. We do not assume any special separability of the two cast parts, and allow parts of arbitrary genus. The running time of our algorithm for determining the castability of a polyhedral object with a given pair of directions is $O(n^2 \log n)$, where n is the combinatorial complexity of the polyhedron.

All the results for opposite cast parts removal in [7, 30, 34] rely on the property that an object is castable if its boundary surface is completely visible from the two opposite removal directions. This is not true when the removal directions are non-opposite: there are polyhedra whose whole boundary is visible from the removal directions but which are not castable with respect to those directions [7]. Consider the polyhedron $\mathcal{P}$ and the removal directions depicted in Figure 5.3. The shaded facets of $\mathcal{P}$ together with the bottom facets form the blue cast part of the boundary; the remaining facets form the red

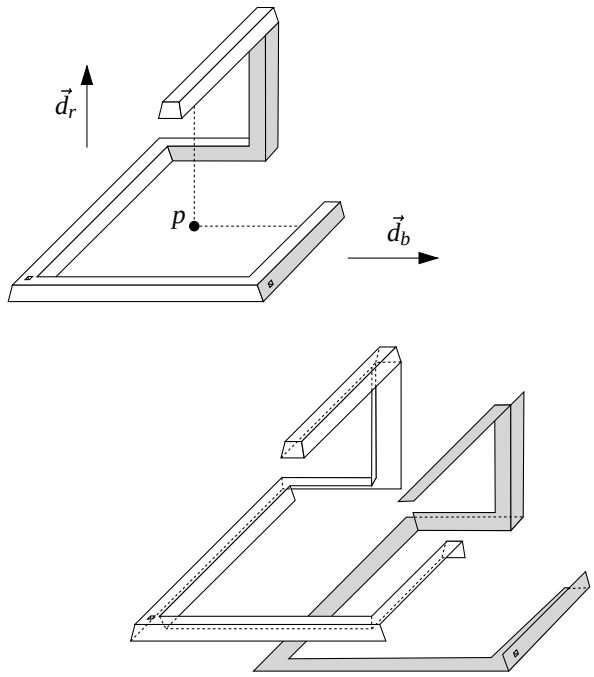


Figure 5.3: The whole boundary is visible from the removal directions but which are not castable with respect to those directions

cast part. Both the red and the blue cast part are terrains in their respective removal directions. There is no good cast, however, for these removal directions: the point p will intersect the interior of $\mathcal{P}$ both when it is moved in direction $\vec{d}_b$ and when it is moved in direction $\vec{d}_r$, so it can neither be in the blue nor in the red cast part. This means that there is no good cast for the directions $\vec{d}_r$ and $\vec{d}_b$. By poking two thin holes into the object, as in Figure 5.3, we can ensure there cannot be a good cast for any other pair of directions either.

For completeness, we also give an $O(n^{14}\log n)$-time algorithm for finding all combinatorially distinct feasible pairs of removal directions. Though the running time is polynomial, the algorithm is clearly of theoretical interest only.

5.2 A characterization of castability

We call an object $\mathcal{Q}$ *castable* with respect to a pair of direction $(\vec{d}_r, \vec{d}_b)$ if we can translate the red cast part $\mathcal{C}_r$ to infinity in direction $\vec{d}_r$ without collision with $\mathcal{Q}$ and int$(\mathcal{C}_b)$, and

then translate the blue cast part $\mathcal{C}_b$ to infinity in direction $\vec{d}_b$ without collision with $\mathcal{Q}$. The order of removal is important.

Imagine that we illuminate $\mathcal{Q}$ with two sources of parallel light. The red light source is at infinity in direction $\vec{d}_r$ and the blue light source is at infinity in direction $\vec{d}_b$. We say that a point p in space is illuminated by red light if a red ray from the red light source can reach p without intersecting $\mathcal{Q}$. The definition for a point p being illuminated by blue light is similar. Note that we assume that a light ray will not stop when it grazes the boundary of $\mathcal{Q}$.

We denote by B an axis-parallel box whose boundary is the outer shape of the cast. There is a (possibly disconnected) subset of $B \setminus \mathcal{Q}$ not illuminated by red light. We call it the *red shadow volume* and denote it by $\mathcal{V}_r$. Similarly, there is a subset of $B \setminus \mathcal{Q}$ not illuminated by blue light. We call it the *blue shadow volume* and denote it by $\mathcal{V}_b$. Note that the object $\mathcal{Q}$ is a closed set and $(\mathcal{C}_r \cup \mathcal{C}_b)$ is an open set. If we sweep $\mathcal{V}_b$ to infinity in direction $\vec{d}_r$, then we will encounter a set of points in B and we denote this set of points by $\mathcal{V}_b^*$. Note that $\mathcal{V}_b^*$ includes $\mathcal{V}_b$ itself.

Lemma 15 *If $\mathcal{Q}$ is castable, then $\mathcal{V}_r \subseteq \mathcal{C}_b$ and $\mathcal{V}_b^* \subseteq \mathcal{C}_r$.*

Proof: By definition, $B \setminus \mathcal{Q}$ is contained in $\mathcal{C}_r \cup \mathcal{C}_b$ and so both $\mathcal{V}_r$ and $\mathcal{V}_b$ are contained in $\mathcal{C}_r \cup \mathcal{C}_b$. Take any point p in $\mathcal{V}_r$. If we move p in direction $\vec{d}_r$ to infinity, then p will be stopped by $\mathcal{Q}$ as p does not receive any red light. So p cannot be a point in the red cast part $\mathcal{C}_r$. Thus, $\mathcal{V}_r \subseteq \mathcal{C}_b$. By similar analysis, $\mathcal{V}_b \subseteq \mathcal{C}_r$. Since $\mathcal{Q}$ is castable, $\mathcal{C}_r$ can be translated first to infinity in $\vec{d}_r$ without colliding with $\mathcal{Q}$ and int$(\mathcal{C}_b)$. Since $\mathcal{V}_b \subseteq \mathcal{C}_r$, we conclude that $\mathcal{V}_b^* \subseteq \mathcal{C}_r$. □

We are now ready to prove the necessary and sufficient condition for an object to be castable.

Theorem 9 *Given an object $\mathcal{Q}$, $\mathcal{Q}$ is castable if and only if $\mathcal{V}_r$ lies in one connected component of $B \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$.*

Proof: First, we prove that the condition is necessary. Since $\mathcal{Q}$ is castable, $\mathcal{V}_b^* \subseteq \mathcal{C}_r$ and $\mathcal{V}_r \subseteq \mathcal{C}_b$ by Lemma 15. Since $\mathcal{C}_b = B \setminus (\mathcal{C}_r \cup \mathcal{Q})$, we have $\mathcal{V}_r \subseteq \mathcal{C}_b = B \setminus (\mathcal{C}_r \cup \mathcal{Q}) \subseteq B \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$. Therefore, if $\mathcal{V}_r$ does not lie in one connected component of $B \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$, then $\mathcal{C}_b$ does not either. This implies that $\mathcal{C}_b$ is not connected, a contradiction.

Second, we prove the sufficiency of the condition. Without loss of generality, let $\vec{d}_r$ be the positive vertical direction. Let f be the top facet of B. Let M be a thin layer of B below f and above $\mathcal{Q}$. There exists a layer M as $\mathcal{Q}$ lies strictly in the interior of B. Let R be the connected component of $B \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$ that contains $\mathcal{V}_r$. Define the blue cast part $\mathcal{C}_b$ to be $R \setminus M$. Then define the red cast part $\mathcal{C}_r$ to be $B \setminus (\mathcal{Q} \cup \mathcal{C}_b)$.

We first argue that each cast part is connected. Since R is connected, we can ensure that $\mathcal{C}_b$ is connected by adjusting the thickness of M. Since both $\mathcal{C}_r$ and M are completely illuminated by red light, every connected component of $\mathcal{C}_r$ overlaps with M and hence $\mathcal{C}_r$ is connected.

We now show that the cast parts can be removed in order. Since $\mathcal{C}_r$ is completely illuminated by red light, $\mathcal{C}_r$ can be translated to infinity in direction $\vec{d}_r$ without colliding with $\mathcal{Q}$. We also claim that this translation of $\mathcal{C}_r$ cannot be obstructed by $\mathcal{C}_b$. Otherwise, a point p in $\mathcal{C}_r$ can see a point q in $\mathcal{C}_b$ in direction $\vec{d}_r$. If the line segment pq contains a point in $\mathcal{V}_b^*$, then q also belongs to $\mathcal{V}_b^*$. Since $\mathcal{C}_b \cap \mathcal{V}_b^* = \emptyset$, q does not belong to $\mathcal{C}_b$, a contradiction. If the line segment pq does not contain any point in $\mathcal{V}_b^*$, then pq lies in $B \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$. Since $q \in \mathcal{C}_b$, p also belongs to the connected component of $B \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$ containing $\mathcal{V}_r$. Thus, $p \in \mathcal{C}_b$, a contradiction.

After removing $\mathcal{C}_r$, $\mathcal{C}_b$ can be removed to infinity in $\vec{d}_b$ without colliding with $\mathcal{Q}$ because $\mathcal{C}_b$ does not contain any point in $\mathcal{V}_b$ by definition. □

The condition in Theorem 9 also implies that $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. However, unlike the case where the two removal directions are opposites of each other, the emptiness of $\mathcal{V}_r \cap \mathcal{V}_b$ does not guarantee castability. Figure 5.4 shows an object that is not castable, even though $\mathcal{V}_r \cap \mathcal{V}_b$ is empty.

Depending on the casting technology used, the construction used in Theorem 9 may not result in useful casts, since it generates walls between $\mathcal{C}_r$ and $\mathcal{C}_b$ that are parallel to $\vec{d}_r$ or $\vec{d}_b$. Sometimes these are unavoidable, but it would be nice to have a method that does not create them if they are not necessary. Let $\mathcal{V}$ be $\mathcal{C}_b \setminus \mathcal{V}_r$. One might try to exploit that $\mathcal{V}$ can also be translated in $\vec{d}_r$ without colliding with $\mathcal{Q}$. Thus, parts of $\mathcal{V}$ may be included in $\mathcal{C}_r$ to remove walls parallel to $\vec{d}_r$ or $\vec{d}_b$. Some walls parallel to $\vec{d}_b$ (resp. $\vec{d}_r$) between $\mathcal{C}_b$ (resp. $\mathcal{C}_r$) and the object can also be removed by doing this.

5.3 Feasibility test for a polyhedron

In this section, we present an $O(n^2 \log n)$-time algorithm for testing the feasibility of a pair of removal directions $(\vec{d}_r, \vec{d}_b)$ for a given polyhedron $\mathcal{P}$. Throughout this section, we assume that $\vec{d}_r$ is the positive vertical direction and that $\mathcal{P}$ lies above the xy-plane. We use $\ell(p)$ to denote the vertical line through a point p.

The *red shadow*, denoted by $\mathcal{S}_r$, is the complement of the set of points on $\partial\mathcal{P}$ that can be illuminated by red light from infinity in direction $\vec{d}_r$ without being obscured by $\mathrm{int}(\mathcal{P})$. The *blue shadow*, denoted by $\mathcal{S}_b$, is the complement of the set of points on $\partial\mathcal{P}$ that can be illuminated by blue light from infinity in direction $\vec{d}_b$ without being obscured by $\mathrm{int}(\mathcal{P})$. Their intersection $\mathcal{S}_r \cap \mathcal{S}_b$ is called the *black shadow*.

For each polyhedron edge e, let $h_b(e)$ denote the plane through e and parallel to $\vec{d}_b$. Then

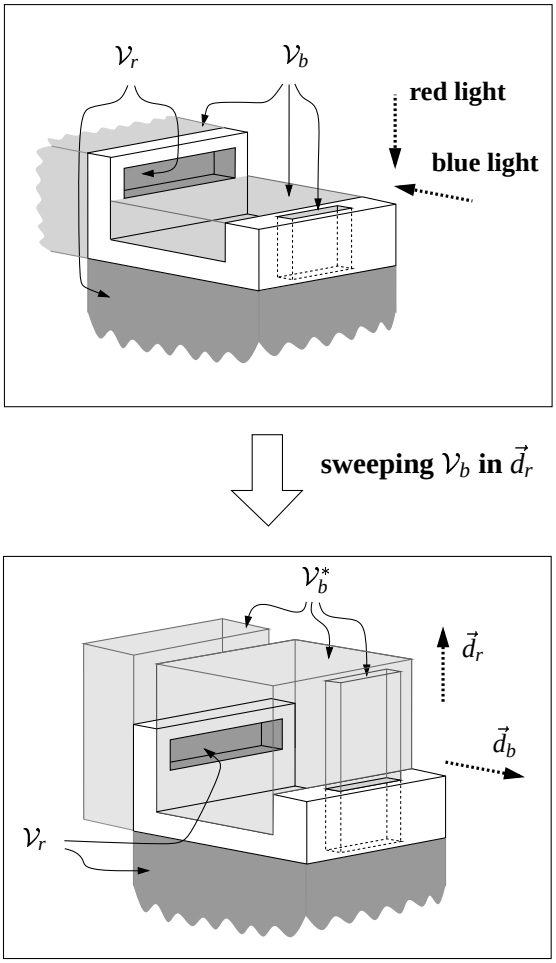


Figure 5.4: An object $\mathcal{Q}$ and its shadow volumes. $\mathcal{V}_r$ intersects two connected components of $B \setminus \mathcal{V}_b^* \cup \mathcal{Q}$.

e is a *blue silhouette edge* if it satisfies two requirements. The first requirement is that the two facets incident to e lie in a closed halfspace bounded by $h_b(e)$ and the dihedral angle through $\text{int}(\mathcal{P})$ is less than π. The second requirement is that if a facet incident to e is parallel to $\vec{d}_b$, then e should be behind that facet when viewing from infinity in direction $\vec{d}_b$. A *lower blue silhouette edge* is a blue silhouette edge e where $\mathcal{P}$ lies *above* $h_b(e)$ locally at e. Similarly, an *upper blue silhouette edge* is a blue silhouette edge e where $\mathcal{P}$ lies *below* $h_b(e)$ locally at e.

For each lower blue silhouette edge e, imagine that e is a neon tube shooting blue rays in

direction $-\vec{d}_b$. We trace the "sheet" of blue rays emanating from e until they hit $\text{int}(\mathcal{P})$, or hit an edge or facet parallel to $\vec{d}_b$ and below $\text{int}(\mathcal{P})$ locally, or reach infinity in direction $-\vec{d}_b$. The union of these intercepted or unintercepted blue rays defines a subset of the plane $h_b(e)$ called a *lower blue curtain*. Note that a lower blue curtain may pass through a facet of $\mathcal{P}$ parallel to $\vec{d}_b$. Such a facet must then be locally above $\text{int}(\mathcal{P})$.

For each upper blue silhouette edge e, we define an *upper blue curtain* symmetrically. We trace the "sheet" of blue rays emanating from e until they hit $\text{int}(\mathcal{P})$, or hit an edge or facet parallel to $\vec{d}_b$ and above $\text{int}(\mathcal{P})$ locally, or reach infinity in direction $-\vec{d}_b$. The union of these intercepted or unintercepted blue rays forms an upper blue curtain. Note that an upper blue curtain may pass through a facet of $\mathcal{P}$ parallel to $\vec{d}_b$. Such a facet must then be locally below $\text{int}(\mathcal{P})$.

Given a blue silhouette edge e, we use $\Gamma(e)$ to denote the blue curtain defined by e. If $\Gamma(e)$ is nonempty, then it is bounded by a silhouette edge e called the *head*, two edges parallel to $\vec{d}_b$ and incident to the endpoints of e called the *side edges*, edges parallel to $\vec{d}_b$ but not incident to the endpoints of e called the *finger edges* and a set $\xi(e)$ of polygonal chains opposite to e called the *tail*. Note that the head and tail of a blue curtain lie on $\partial\mathcal{P}$

We divide castability testing into three steps. We first verify that the boundary of $\mathcal{P}$ is completely illuminated by red and blue light. That is, $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. Once this test is passed, we then check whether $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. If this test is passed, then we construct the red and blue cast parts and verify that they are connected.

5.3.1 Testing emptiness of the black shadow

The emptiness of $\mathcal{S}_r \cap \mathcal{S}_b$ can be tested in $O(n^2 \log n)$ time as follows. We put a horizontal plane H above $\mathcal{P}$ and compute the projection of $\mathcal{P}$ onto H with the hidden portion removed. The resulting arrangement is known as the visibility map. We project this visibility map vertically downward on the boundary of $\mathcal{P}$. This tells us which part of $\partial\mathcal{P}$ is illuminated by red light. An edge in the visibility map is the projection of a polyhedron edge. A vertex in the visibility map is the projection of a polyhedron vertex or the intersection between the projections of two polyhedron edges. Clearly, the size of the visibility map is $O(n^2)$. It can be computed in $O(n^2 \log n)$ time using a plane sweep over the projection of all polyhedron edges to remove the hidden line segments. Output-sensitive algorithms for visibility map computation are also known [1]. Thus, determining the parts of $\partial\mathcal{P}$ illuminated by red light can be done in $O(n^2 \log n)$. Similarly, we can determine the parts of $\partial\mathcal{P}$ illuminated by blue light in $O(n^2 \log n)$. We can then decide whether $\mathcal{S}_r \cap \mathcal{S}_b$ is empty by testing the intersection separately on every facet of $\partial\mathcal{P}$, for instance with a plane sweep algorithm. In total, this test takes time $O(n^2 \log n)$.

5.3.2 Testing emptiness of the black shadow volume

Once we know that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty, we can determine if the black shadow volume is empty by examining the lower envelope, denoted by $\mathcal{E}$, of blue shadow facets and lower blue curtains. This is more efficient than computing $\mathcal{V}_r \cap \mathcal{V}_b$ directly. We show how this is done in the following.

Lemma 16 *Let $\mathcal{E}^*$ be the set of points in B encountered while we sweep $\mathcal{E}$ to infinity vertically upward. Then $\mathcal{E}^* = \mathrm{cl}(\mathcal{V}_b^*)$.*

Proof: Let p be a point in $\mathcal{E}^*$, and let q be the point $\ell(p) \cap \mathcal{E}$. By definition, q is either on a blue shadow facet or a lower blue curtain. Therefore q is in $\mathrm{cl}(\mathcal{V}_b)$ and p is in $\mathrm{cl}(\mathcal{V}_b^*)$.

Let q be a point in $\mathrm{cl}(\mathcal{V}_b^*)$, and let p be the lowest point of $\ell(q) \cap \mathrm{cl}(\mathcal{V}_b^*)$. By definition, p is on a facet σ of $\mathrm{cl}(\mathcal{V}_b)$ which bounds $\mathrm{cl}(\mathcal{V}_b)$ from below. Since $\mathrm{cl}(\mathcal{V}_b)$ is bounded by blue shadow facets or blue curtains, σ is either a blue shadow facet or a lower blue curtain. Therefore p is in $\mathcal{E}$ and q is in $\mathcal{E}^*$. □

The emptiness of the black shadow volume is now determined by the necessary and sufficient condition stated in Lemma 18. We first need a technical lemma.

Lemma 17 *Suppose that $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. Then for any two blue silhouette edges e and f, and points $p \in e$ such that $\ell(p) \cap \mathrm{int}(\Gamma(f)) \neq \emptyset$, p is not above $\ell(p) \cap \Gamma(f)$.*

Proof: Assume to the contrary that there is a point $p \in e$ such that p is above $\ell(p) \cap \Gamma(f)$. We can shift $\ell(p)$ slightly to another vertical line ℓ such that ℓ stabs the interior of $\Gamma(f)$ and a facet incident to e. Thus, ℓ intersects $\mathrm{int}(\mathcal{P})$ above $\ell \cap \Gamma(f)$. If $\Gamma(f)$ is an upper blue curtain, then this implies that a short segment on ℓ below $\ell \cap \Gamma(f)$ does not receive red or blue light. If $\Gamma(f)$ is a lower blue curtain, then a short segment above $\ell \cap \Gamma(f)$ does not receive red or blue light. In either case, $\mathcal{V}_r \cap \mathcal{V}_b$ would be nonempty, a contradiction. □

Lemma 18 *Suppose that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. Then $\mathcal{V}_r \cap \mathcal{V}_b$ is non-empty if and only if for two lower blue silhouette edges e and f, and for some point $p \in e$ such that $\ell(p) \cap \mathrm{int}(\Gamma(f)) \neq \emptyset$ or $\ell(p) \cap \mathrm{int}(\xi(f)) \neq \emptyset$, p is not below $\ell(p) \cap \Gamma(f)$.*

Proof: We prove sufficiency first. Suppose that $p \in e$ is a point such that $\ell(p) \cap \mathrm{int}(\Gamma(f)) \neq \emptyset$ and p is not below $\ell(p) \cap \Gamma(f)$. By Lemma 17, p is also not above $\ell(p) \cap \Gamma(f)$. Thus, $p = \ell(p) \cap \Gamma(f)$. The interior of $\Gamma(f)$ would contain the point p. By definition, no boundary point of $\mathcal{P}$ below $\mathrm{int}(\mathcal{P})$ locally can lie in the interior of a blue curtain, a contradiction.

The remaining alternative is that $\ell(p)$ intersects $\xi(f)$ where p is not below $\ell(p) \cap \Gamma(f)$. If we shift $\ell(p)$ slightly in direction $\vec{d}_b$ to a vertical line ℓ, we claim that ℓ must intersect

the interior of a facet σ incident to e. Otherwise, since e is a lower blue silhouette edge, a facet incident to e would face downward and direction $-\vec{d}_b$, and so this facet would contain a point in black shadow, a contradiction. Observe that ℓ also intersects the interior of $\Gamma(f)$. By definition, the interior of σ cannot lie on $\text{int}(\Gamma(f))$ since $\mathcal{P}$ is above σ locally. This implies that there is a short segment on ℓ above $\text{int}(\Gamma(f))$ that does receive neither red nor blue light, which contradicts the emptiness of $\mathcal{V}_r \cap \mathcal{V}_b$.

We now prove necessity. Since $\mathcal{S}_r \cap \mathcal{S}_b$ is empty, any facet of $\text{cl}(\mathcal{V}_r \cap \mathcal{V}_b)$ is parallel to $\vec{d}_r$ or $\vec{d}_b$. At least one facet σ of $\text{cl}(\mathcal{V}_r \cap \mathcal{V}_b)$ is parallel to $\vec{d}_b$ and bounds $\text{cl}(\mathcal{V}_r \cap \mathcal{V}_b)$ from below, otherwise $\text{cl}(\mathcal{V}_r \cap \mathcal{V}_b)$ would be unbounded. The facet σ cannot receive any red light as it bounds the black shadow volume. Thus, σ must receive some blue light, otherwise σ would be in black shadow which is supposed to be empty. Therefore, σ must lie on some lower blue curtain $\Gamma(f)$. Let z be a point in $\text{int}(\sigma)$. If we shoot a ray upward from z, the ray hits $\partial\mathcal{P}$ at a point $v(z)$. Suppose that we move z in the direction $-\vec{d}_b$. The height of $v(z)$ from $\Gamma(f)$ is monotonically decreasing and remains non-negative. Otherwise, there would be a position such that $v(z)$ becomes a point in the black shadow which is impossible (See Figure 5.5 (a).) Before or just when $\ell(z)$ stabs an edge g of $\xi(f)$, $v(z)$ reaches an edge e such that $\mathcal{P}$ lies locally on one side of a vertical plane through e. Otherwise, a facet adjacent to g would contain a point in black shadow, a contradiction (See Figure 5.5 (b).) Observe that e is also a lower blue silhouette edge, and $\ell(z) \cap e$ does not lie below $\ell(z) \cap \Gamma(f)$ (See Figure 5.5 (c).) Clearly, $\ell(v(z)) = \ell(z)$ either intersects the interior of $\Gamma(f)$ or $\xi(f)$, and $v(z)$ is not below the intersection point. □

To test the condition in Lemma 18, we identify all lower blue silhouette edges and construct the lower blue curtains. Then we identify all blue shadow facets and construct $\mathcal{E}$, the lower envelope of all lower blue curtains and all blue shadow facets. While we construct $\mathcal{E}$ we can check whether $\mathcal{V}_r \cap \mathcal{V}_b$ is empty.

We show below that the lower envelope $\mathcal{E}$ formed by all lower blue curtains and all blue shadow facets has $O(n^2)$ complexity. We first need a technical lemma.

Lemma 19 *If $\mathcal{S}_r \cap \mathcal{S}_b$ is empty, then for any blue shadow facets f_1 and f_2, and points $p \in \text{int}(f_1)$, $\ell(p) \cap \text{int}(f_2)$ is empty.*

Proof: Assume to the contrary that there is a point $p \in \text{int}(f_1)$ such that $\ell(p) \cap \text{int}(f_2)$ is not empty. Without loss of generality, assume that p is below $\ell(p) \cap f_2$. Since $f_2 \subseteq \partial\mathcal{P}$, all points of $\ell(p)$ lying below $\ell(p) \cap f_2$ can not get any red light. Thus, p is a black point which contradicts emptiness of $\mathcal{S}_r \cap \mathcal{S}_b$. □

Lemma 20 *Suppose that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. Then the lower envelope $\mathcal{E}$ formed by all lower blue curtains and all blue shadow facets has complexity $O(n^2)$.*

Proof: We have three different kinds of edges: shadow facet edges including heads and tails of lower blue curtains, side edges and finger edges. The complexity of $\mathcal{E}$ is deter-

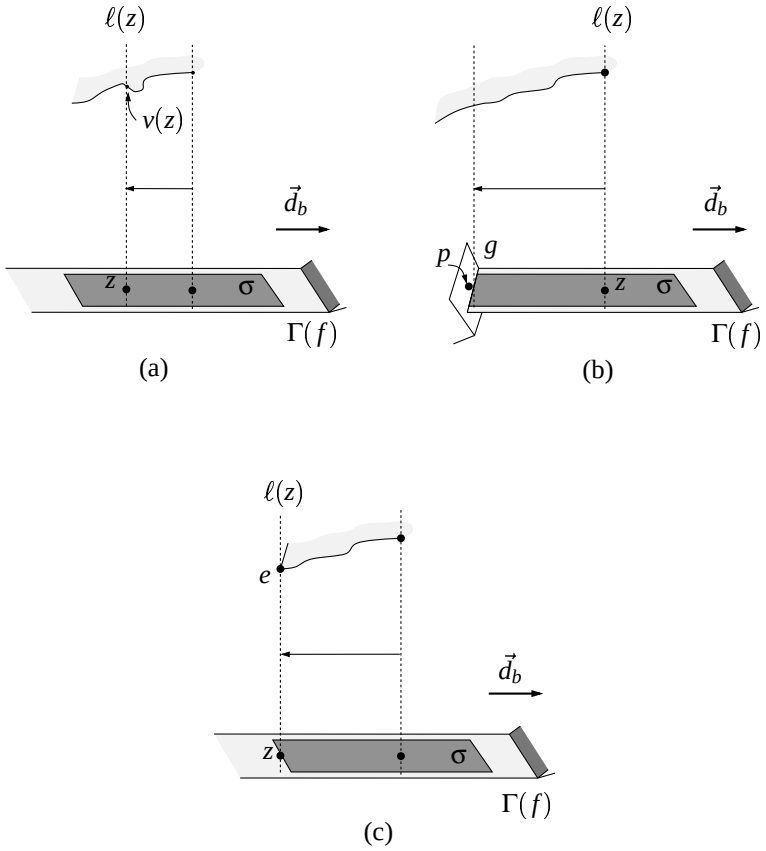


Figure 5.5: (a) $v(z)$ in the black shadow, (b) A point p in the black shadow, and (c) A lower silhouette edge e, where $\ell(z) \cap e$ does not lie below $\ell(z) \cap \Gamma(f)$

mined by the number of these edges and new vertices generated by these edges.

By Lemma 19, shadow facet edges do not cross each other in the projection, that is, they do not introduce any new vertex in $\mathcal{E}$.

Now consider a finger edge of a lower blue curtain $\Gamma(e)$. It can be divided into $O(n)$ segments of two types : segments lying on facets of $\mathcal{P}$ which are parallel to $\vec{d}_b$, and segments lying on the other blue curtains. Figure 5.6 shows segments of each type. The segments of the former type are shadow facet edges and there are $O(n^2)$ of them. So we only consider the segments of the latter type. A segment of the latter type is the intersection of $\Gamma(e)$ and another blue curtain, say $\Gamma(f)$. If $\Gamma(f)$ is a lower blue curtain, then one of two facets σ incident to f is a red shadow facet. Since a lower blue curtain $\Gamma(e)$ intersect σ, σ contains a black point, which contradicts $\mathcal{S}_r \cap \mathcal{S}_b = \emptyset$. So $\Gamma(f)$ must

be an upper blue curtain. Since points in the interior of upper blue curtains do not appear in $\mathcal{E}$, we conclude that finger edges do not introduce any new vertex in $\mathcal{E}$.

Now we only need to check how many new vertices are generated by side edges and shadow facet edges. Let e be a side edge of a lower blue curtain and h be a vertical plane containing e. Then h intersects a shadow facet f_i in a line segment, denoted by s_i. Since a shadow facet edge is either an edge of $\mathcal{P}$ or the projection of an edge of $\mathcal{P}$ on $\partial\mathcal{P}$ in $-\vec{d}_b$ direction, h intersects $O(n)$ shadow facets in $O(n)$ nonintersecting line segments. Now it becomes a 2-dimensional problem of computing lower envelope of s_i's and e on h. The lower envelope of s_i's and e has linear complexity. Since $\mathcal{P}$ has $O(n)$ side edges, $\mathcal{E}$ has $O(n^2)$ vertices in total. □

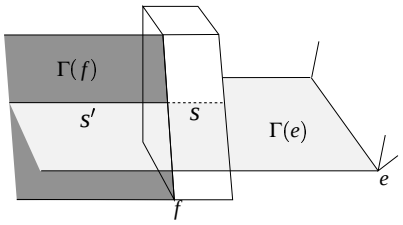


Figure 5.6: A finger edge consisting of two segments : s lying on a facet parallel to $\vec{d}_b$, and s' lying on $\Gamma(f)$.

Now we show below how to construct $\mathcal{E}$ in $O(n^2 \log n)$ time if $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. In the process, we also verify whether $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. This implies that we can construct the swept volume $\mathcal{V}_b^*$ in time $O(n^2 \log n)$ by computing this lower envelope.

Lemma 21 *Suppose that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. Then we can test emptiness of $\mathcal{V}_r \cap \mathcal{V}_b$, and if $\mathcal{V}_r \cap \mathcal{V}_b$ is empty we can construct $\mathcal{E}$ in $O(n^2 \log n)$ time.*

Proof: To construct $\mathcal{E}$, we first project all shadow facets $\mathcal{S}_b$ on a horizontal plane h, and partition h into slabs by drawing lines parallel to $\vec{d}_b$ through all projected vertices of $\mathcal{P}$. All blue shadow facets can be identified in $O(n^2 \log n)$ time by computing the complement of the part of $\partial\mathcal{P}$ which is illuminated by blue light. Edges of blue shadow facets are edges or parts of edges of $\mathcal{P}$, or edges in the tails of blue curtains. Since a vertical plane parallel to $\vec{d}_b$ intersects $O(n)$ edges of $\mathcal{P}$ and $O(n)$ edges in the tails of blue curtains, the above construction results in a subdivision of complexity $O(n^2)$.

A slab consists of two types of regions : the projection of blue shadow facets, and the rest including two unbounded regions. We call regions of the latter type "empty". We label these regions in sorted order in $-\vec{d}_b$, that is, the region unbounded to infinity in $\vec{d}_b$ is Δ_1, the next is Δ_2, and so on. We denote the boundary between two consecutive regions Δ_i and Δ_{i-1} by ζ_i. Each Δ_i belongs to a projected shadow facet or is empty.

The lower blue silhouette edges can be identified in $O(n)$ time. To construct its lower blue curtain we intersect a plane with $\mathcal{P}$ in $O(n \log n)$ time so we can construct all lower blue curtains in time $O(n^2 \log n)$ time. For each slab, we traverse each region starting from Δ_1 and maintain a dictionary [49] of lower blue curtains ordered by their heights.

At a region Δ_i, other than Δ_1, we first identify all new lower blue curtains, and then we test the emptiness of $\mathcal{V}_r \cap \mathcal{V}_b$. We pick a point p in $\text{int}(\zeta_i)$. If p lies in the projection of some lower blue silhouette edges, then we insert all the curtains of these edges to the dictionary, and by Lemma 18, we test whether $\mathcal{V}_r \cap \mathcal{V}_b$ is empty as follows: If ζ_i is the projection of lower blue silhouette edges, then let e be the lowest one among them. We pick a point q in e of which vertical projection is in $\text{int}(\zeta_i)$, and shoot a ray downward from q. If there is any lower blue curtains below q then $\mathcal{V}_r \cap \mathcal{V}_b$ is not empty. Once we find $\mathcal{V}_r \cap \mathcal{V}_b$ is not empty, we stop and report that $\mathcal{P}$ is not castable. Otherwise, if p lies on the projection of edges in tails of lower blue curtains, then we delete all these curtains from the dictionary.

After updating the dictionary, we shoot a ray upward from p, and report the lowest curtain hit by the ray. This can be done in $O(\log n)$ time by querying the dictionary. Then we fill the region with the projection of the lowest curtain. Figure 5.7 shows part of a slab consisting of two blue shadow regions and four curtain regions. At ζ_{i-1}, we insert $\Gamma(f)$ to the dictionary. Now the dictionary has $\Gamma(e)$ and $\Gamma(f)$. Then we fill Δ_{i-1} with the projection of $\Gamma(f)$. Then at ζ_i, we delete $\Gamma(f)$ from the dictionary. At ζ_{i+1}, we shoot a ray upward, find the $\Gamma(e)$ as the lowest curtain, and fill Δ_{i+1} with the projection of $\Gamma(e)$.

There are $O(n)$ slabs, and each slab has $O(n)$ regions. There are $O(n)$ lower blue curtains and for each slab, each curtain is inserted into and deleted from the dictionary at most once in $O(\log n)$ time. So the total dictionary update time for all slabs is $O(n^2 \log n)$. At each region we identify the lowest curtain and test the emptiness of black shadow volume in $O(\log n)$ time. Thus we can test the the emptiness of black shadow volume in $O(n^2 \log n)$ time in total. If black shadow volume is empty, we have obtained in $O(n^2 \log n)$ time the portion of the lowest blue curtain above each empty region in each slab. They form $\mathcal{E}$ together with the blue shadow facets. □

5.3.3 Cast part construction

Finally, we show how to construct the red and blue cast parts. In the process, we also verify whether the cast parts are connected.

Points on blue shadow facets and points close to and above lower blue curtains are not illuminated by blue light. So they can only be removed in direction $\vec{d}_r$. Other points encountered while translating these points towards infinity in $\vec{d}_r$ should then belong to $\mathcal{C}_r$ too. This is exactly the subset $\mathcal{E}^*$ of B swept by the lower envelope $\mathcal{E}$ of lower blue curtains and blue shadow facets in direction $\vec{d}_r$. $\mathcal{E}^*$ may be disconnected. Since $\mathcal{P}$ is strictly contained in B, we can take a layer of material M beneath the top facet of B and above $\mathcal{P}$ and use M to connect all the components in $\mathcal{E}^*$. By Lemma 21, we can compute

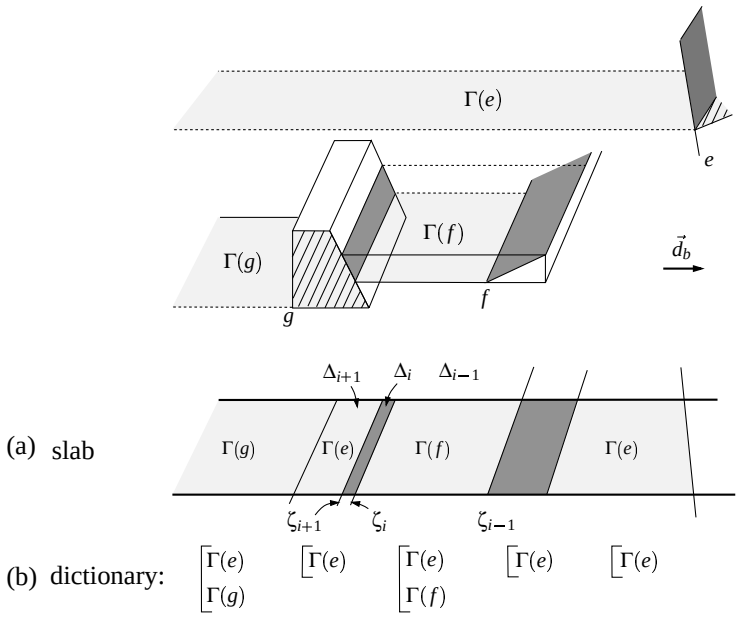


Figure 5.7: (a) A part of a slab, and (b) Content of the dictionary at each region

$\mathcal{E}$ in $O(n^2 \log n)$ time and so $\mathcal{E}^* \cup M$ can then be computed in the same time. $\mathcal{E}^* \cup M$ is our potential red cast part. All the points in $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ are removable in direction $\vec{d}_b$. So $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ is our potential blue cast part. However, $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ may be disconnected. Thus, we will try to attach some components in $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ to $\mathcal{E}^* \cup M$ instead.

Such a process is guided by the condition in Theorem 9. Observe that $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ is a subset of $B \setminus (\mathcal{V}_b^* \cup \mathcal{P})$. From the above analysis, any blue cast part and hence $\mathcal{V}_r$ lies inside $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$. Thus, we can attach every component of $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ not containing any point in $\mathcal{V}_r$ to $\mathcal{E}^* \cup M$. These components are removable in direction $\vec{d}_r$ as they do not contain points in $\mathcal{V}_r$. In addition, if there are more than one remaining component of $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ containing $\mathcal{V}_r$, then we can abort and report that $\mathcal{P}$ is not castable. Otherwise, we have the cast parts.

It is unnecessary to compute $\mathcal{V}_r$. Every facet bounding $\mathcal{V}_r$ is connected to some red shadow facet. The red shadow facets can be computed in $O(n^2 \log n)$ time using visibility maps. Each red shadow facet lies on a facet bounding $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$. We identify the set of facets bounding $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ that contain the red shadow. Then we test whether this set of facets lie in the same component of $\text{cl}(B \setminus (\mathcal{E}^* \cup M))$ using a linear-time graph

traversal.

Theorem 10 *Let $\mathcal{P}$ be a simple polyhedron with n vertices. Given a pair of directions, we can determine castability and construct cast parts, if castable, of $\mathcal{P}$ in $O(n^2 \log n)$ time and $O(n^2)$ space. The cast parts constructed have $O(n^2)$ complexity.*

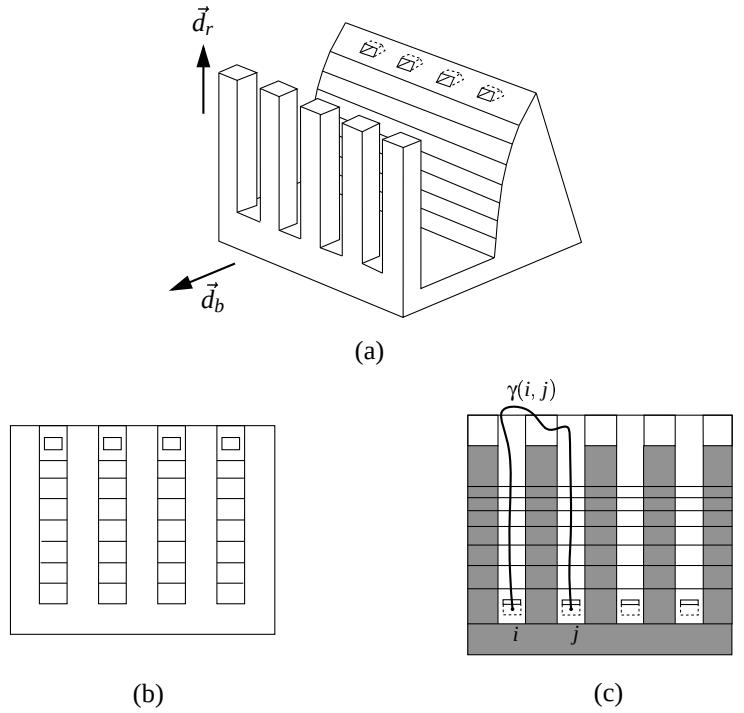


Figure 5.8: (a) A polyhedron with five vertical legs and four small holes, (b) The visibility map from infinity in direction $\vec{d}_b$, and (c) The visibility map from infinity in direction $\vec{d}_r$

The optimality of the bound follows from the example in Figure 5.8. The polyhedron shown in Figure 5.8 (a) has a row of vertical "legs". Behind these legs, there is a row of small holes such that parts of the cast inside the holes can only be removed in direction $\vec{d}_b$. The polyhedron has seven thin rectangular facets below the holes. Figure 5.8 (b) shows the visibility map from the blue light source. If there are $n/8$ vertical legs and $n/8$ horizontally long retangles, then the map has $\Theta(n^2)$ complexity, which shows that $\mathcal{V}_b$ has $\Theta(n^2)$ complexity. It follows that the lower envelope $\mathcal{E}$ of all the lower curtains and blue shadow has complexity $\Omega(n^2)$. This shows that the analysis of the cast part size of our construction is tight. Figure 5.8 (c) shows a top view of the polyhedron with the blue shadow.

Since parts of the cast inside the holes are red shadow volumes, they need to be in the

blue cast part, which is one connected component. It is clear that the blue shadow on $\partial\mathcal{P}$ must belong to $\mathcal{C}_r$. Thus, in any cast construction, there exists a path $\gamma(i,j)$ connecting any pair of the red shadow volumes i, j inside the holes through $\text{int}(\mathcal{C}_b)$. Figure 5.8 (c) shows a path $\gamma(i,j)$ connecting red shadow volumes i, j inside the holes. Let $\gamma^*(i,j)$ be the projection of the path on $\partial\mathcal{P}$ in $-\vec{d}_r$. Then all points in $\gamma^*(i,j)$ belong to the blue cast part. Otherwise , there is a point $p \in \mathcal{C}_r$ in $\gamma^*(i,j)$ which implies that the point $\ell(p) \cap \gamma(i,j)$ would belong to $\mathcal{C}_r$, a contradiction. For a similar reason, $\gamma^*(i,j)$ cannot intersect the blue shadow of the object. Note that $\gamma^*(i,j)$ intersects all the thin rectangular facets below the holes.Therefore, the red and blue cast parts intersect the sequence of thin rectangular facets alternately, which results in $\Omega(n^2)$ complexity of the cast part. It follows that the size of the cast produced by our construction is worst-cast optimal.

5.4 Finding a pair of directions

We have seen how to test whether a polyhedron $\mathcal{P}$ is castable in a given pair of directions $(\vec{d}_r, \vec{d}_b)$. In this section we describe an algorithm to solve the following problem: Given a polyhedron $\mathcal{P}$, decide whether there is a pair of directions $(\vec{d}_r, \vec{d}_b)$ in which $\mathcal{P}$ is castable. In fact, we will solve the more general problem of finding all pairs of directions $(\vec{d}_r, \vec{d}_b)$ for which $\mathcal{P}$ can be cast.

The set of all pairs of directions forms a 4-dimensional parameter space Ψ. We choose an appropriate parameterization that gives rise to algebraic surfaces in Ψ, see for instance Latombe's book [35]. Our goal is to compute that part of Ψ that corresponds to pairs of directions in which $\mathcal{P}$ is castable. As we have proven before, castability depends on a number of simple combinatorial properties: the emptiness of the black shadow, the configuration of the curtain projections, and the connectedness of the blue cast part. We will compute an arrangement of algebraic surfaces in Ψ that includes all pairs of directions where one of these properties could possibly change. The following lemma enumerates all relevant situations.

Lemma 22 *Let γ_1 and γ_2 be two pairs of directions, such that $\mathcal{P}$ is castable in γ_1 but not in γ_2. Let π be any path in 4-dimensional configuration space Ψ connecting γ_1 and γ_2. Then on π there is a pair of directions $(\vec{d}_r, \vec{d}_b)$ such that one of the following conditions holds:*

(i) A facet of $\mathcal{P}$ is parallel to $\vec{d}_r$ or $\vec{d}_b$.
(ii) The projection in direction $\vec{d}_r$ of a vertex v coincides with the projection of an edge e. Here edges and vertices are edges and vertices of $\mathcal{P}$ or of the blue shadow $\mathcal{S}_b$.
(iii) Two polyhedron vertices lie in a plane parallel to the plane determined by $\vec{d}_r$ and $\vec{d}_b$.

Proof: As we proved before, castability of $\mathcal{P}$ depends on three factors:

- The black shadow on the surface of $\mathcal{P}$ is empty.
- The projection of the blue curtains forms a legal arrangement.
- The blue cast part is connected.

Let's first consider the black surface shadow: It is empty if and only if the blue surface shadow is completely visible from the red direction. The blue surface shadow changes combinatorially if and only if the visibility map of $\mathcal{P}$ in $\vec{d}_b$ changes. This can happen only when a facet becomes parallel to $\vec{d}_b$, or when a vertex or edge of $\mathcal{P}$ passes in front of another edge or vertex. These possibilities are included in cases (i) and (ii). We now argue that the combinatorial structure of the visibility map of $\mathcal{P}$ *and* the blue shadow in $\vec{d}_r$ can change only if a vertex or edge of $\mathcal{P}$ passes in front of an edge or vertex of either $\mathcal{P}$ or $\mathcal{SH}_b$. Again this is included in case (ii).

If the black surface shadow is empty, it remains to verify whether the projection of the blue curtains and the blue shadow forms a legal configuration. Consider the arrangement of all the blue curtains. This arrangement can only change when the intersection pattern of two curtain changes. The edges in the tails of the curtains are blue shadow edges, so any change in their projections leads to a situation as in case (ii) of the lemma. The only remaining possibility for the intersection pattern of two curtains to change is when the projection of a polytope vertex passes over a polytope edges (case (ii) again), or over the projection of the side edge of a curtain. Since the side edge of a curtain is determined by another vertex of the polytope and the blue direction $\vec{d}_b$, this leads to case (iii).

We have now seen that if none of cases (i), (ii), (iii) happens, the combinatorial structure of the surface shadows and of the projection of the curtains cannot change. Since the red cast part is defined as the volume above the lower envelope of the blue shadow and the blue curtains, it follows that the combinatorial structure of the red cast part cannot change without leading to a configuration as postulated in the lemma. In particular, the combinatorial structure, and therefore the topology, of the union of the object and the red cast part can not change. Since the blue cast part is the complement of these, it cannot change from being connected to being disconnected without a situation as prescribed in the lemma. □

We can now turn this characterization into an algorithm.

Theorem 11 *Given a polyhedral object $\mathcal{P}$ with n vertices and edges, we can in time $O(n^{14}\log n)$ construct a set of all possible pairs of directions in which $\mathcal{P}$ is castable.*

Proof: As mentioned before, we consider a 4-dimensional parameter space, and construct a set of algebraic surfaces. These surfaces correspond to the cases listed in the previous lemma.

Clearly, there are $O(n)$ surfaces for case (i), and $O(n^2)$ surfaces for case (iii). For case (ii), we observe that there are $O(n)$ vertices and edges of $\mathcal{P}$, while there can be $O(n^2)$ edges and vertices of the blue shadow. We create $O(n^2)$ surfaces where an object vertex lies in the plane defined by an object edge and one of the directions $\vec{d}_b$, $\vec{d}_r$. We create $O(n^3)$ surfaces defined by an object edge, an object facet, and an object vertex (the set of all direction pairs where the projection of the vertex along one direction on the facet lies in the projection of the edge along the other direction). Finally, we make $O(n^3)$ surfaces defined by three object edges.

We have now arrived at a set of $O(n^3)$ algebraic surfaces in our 4-dimensional configuration space. All pairs of directions leading to a situation as in the lemma lie on one of the surfaces. Consequently, it is sufficient to sample one configuration in every cell of the arrangement of the surfaces.

The arrangement of $O(n^3)$ surfaces has complexity $O(n^{12})$, and so there are at most $O(n^{12})$ pairs of directions that we can test using the algorithm from the previous section. Since each cell in the arrangement takes $O(n^2 \log n)$ time, we conclude that all directions for which there is a good cast can be computed in $O(n^{14} \log n)$ time. □

Part III

The Reflex-free hull and Cavities

CHAPTER 6

The reflex-free hull

6.1 Introduction

Computational geometers have identified many classes of 2D polygons (convex, star-shaped, L-convex, externally visible, edge-visible, LR-visible, street, person… [50, 54]), but few classes of 3D polyhedra. Perhaps the fact that 3D polyhedra support rich classes of topological structure in the form of knots and links has overshadowed the identification of geometric structure. In applications such as manufacturing or molecular analysis, however, geometric structures such as cavities or docking sites are important.

In the plane, the difference between a simple polygon and its convex hull is a number of simple, polygonal bays, from which one can obtain a natural description of a polygon as a tree of unions and differences of convex pieces [52]. In space, it has been suggested that the same approach be used to define pockets in a search for casting directions [17], but in fact the difference between a polyhedron and its convex hull need not have a natural decomposition, and may have more complicated topology than the original polyhedron [7]: subtracting an object from its convex hull in 3D may leave one component with complex, non-manifold topology. In fact, there are castable polyhedra for which the algorithm of Chen at al. [17] will not find a feasible direction. To our knowledge, we are not aware of previous work in computational geometry concerning the identification of depressions.

In this chapter, we propose a hull operator that allows us to define a 3D analogue to bays in polygons. Section 6.2 defines the notion of *reflex-free sets* and *cavities*. Section 6.3 defines the *reflex-free hull*, *Rfh*, and Section 6.4 establishes some basic results about the *Rfh* of polyhedral sets, including the fact that the *Rfh* has linear complexity even though it allows a rich set of topological types. Section 6.5 shows that the reflex-free hull bounds the limit of a process of filling cavities, but that obtaining it computationally in this man-

ner would be challenging. Finally, Section 6.6 relates the reflex-free hull to other possible hull definitions that either have high complexities or limited topologies.

6.2 Preliminaries

We begin with basic geometric and topological definitions and notation [8, 26] for the sets that we consider in three-dimensional space, $\mathbb{R}^3$. We classify boundary points geometrically, and define reflex-free sets.

A *k-simplex* is the convex hull of $k+1$ affinely independent points. In $\mathbb{R}^3$, we have points, line segments, triangles, and tetrahedra as the 0-, 1-, 2-, and 3-simplices, respectively. The empty set is considered a (-1)-simplex. Notice that the boundary of a simplex is a collection of lower dimensional simplices. A *simplicial complex* is a collection of simplices with disjoint interiors that is closed under the operations of intersection and taking boundaries.

For our purposes in this chapter, a *polyhedron* is the union of the simplices in a finite simplical complex. A *polyhedral set* is homeomorphic to a polyhedron. We restrict our discussion to polyhedral sets to avoid wild topological beasties like the Alexander horned sphere [26]. Section 6.4 further restricts the discussion to polyhedral sets when it investigates combinatorial properties of reflex-free hulls.

A set is *closed* if and only if it contains all of its limit points; the *closure* of a set, $\mathrm{cl}(S)$, is the union of S with its set of limit points. The *complement* of a set $\overline{S} = \mathbb{R}^3 \setminus S$. For any vector $v \in \mathbb{R}^3$, we define the *v-plane* $h_v(p) = \{q \mid (q-p)\cdot v = 0\}$, and the *closed v-halfspace* $h_v^-(p) = \{q \mid (q-p)\cdot v \leq 0\}$. We may suppress subscripts or arguments and write h and h^- when they can be understood from context.

We use the Euclidean metric in $\mathbb{R}^3$, and denote the distance between two points by $d(p,q)$ and between two sets by $d(A,B) = \limsup\{d(a,b) \mid a \in A, b \in B\}$. For $\varepsilon > 0$, the *open* ε*-ball* is $B_\varepsilon(p) = \{q \mid d(p,q) < \varepsilon\}$. The *interior* of a set $\mathcal{Q}$, denoted $\mathrm{int}(\mathcal{Q})$, are the points of $\mathcal{Q}$ for which we can find an $\varepsilon > 0$ such that $B_\varepsilon(p) \subset \mathcal{Q}$. The *boundary* is defined $\partial\mathcal{Q} = \mathcal{Q} \setminus \mathrm{int}(\mathcal{Q})$.

If every boundary point has a neighborhood that is homeomorphic to a half-ball, then $\mathcal{Q}$ is called a *three-manifold with boundary*. We cannot restrict ourselves to manifolds, since non-manfold sets can arise as reflex-free hulls in degenerate configurations. Examples of polyhedral sets that are not three-manifolds include any pair of tetrahedra joined at a vertex or along an edge, and any finite union of one and two dimensional simplices. Points that do not have ball or half-ball neighborhoods are called *singular*.

We classify each boundary point of a polyhedral set, $p \in \mathcal{Q}$, based intuitively on whether $\mathcal{Q}$ or $\overline{\mathcal{Q}}$ can be oriented to hold water at p. Non-manifold sets complicate these definitions. For example, a singular point p appears on the boundary more than once when $\overline{\mathcal{Q}}$ is not

connected in the neighborhood of p. We call a connected component $C \subset B_\varepsilon(p) \cap \overline{\mathcal{Q}}$ an *appearance of p on $\mathcal{Q}$* if $p \in \mathrm{cl}(C)$, and define the *neighborhood of an appearance* as $B_\varepsilon(p) \setminus C$.

For any $\varepsilon > 0$ and vector $v \in \mathbb{R}^3$, we define the *hemisphere* $H_{v,\varepsilon}(p) = B_\varepsilon(p) \cap h_v^-(p) \cap \overline{\{p\}}$. To simplify classification, $H_{v,\varepsilon}(p)$ does not contain p or the boundary points of $B_\varepsilon(p)$. Again, we suppress subscripts or argument when they can be understood from context.

We classify an appearance of point $p \in \partial\mathcal{Q}$ based on the relation of a hemisphere to the neighborhood of the appearance, which we denote $\mathcal{N}$. We say that p appears as a

- *reflex point* if there is a hemisphere at p inside the neighborhood $\mathcal{N}$. That is, if there exists a vector v and $\varepsilon > 0$ such that $H_{v,\varepsilon}(p) \subset \mathrm{int}(\mathcal{N})$.
- *convex point* if there is a hemisphere outside $\mathcal{N}$. That is, if there exists $H_{v,\varepsilon}(p) \subset \overline{\mathcal{N}}$.
- *flat point* if there exists an $\varepsilon > 0$ and $v \in \mathbb{R}^3$ such that $H_{v,\varepsilon}(p) \subset \mathcal{N}$ and $H_{-v,\varepsilon}(p) \subset \mathrm{cl}(\overline{\mathcal{N}})$.
- *nearly reflex point* if p is neither reflex nor flat and there exists $H_{v,\varepsilon}(p) \subset \mathcal{N}$.
- *nearly convex point* if p is neither convex nor flat and there exists $H_{v,\varepsilon}(p) \subset \mathrm{cl}(\overline{\mathcal{N}})$.
- *saddle point* otherwise. That is, for every $\varepsilon > 0$ and vector v, hemisphere $H_{v,\varepsilon}(p)$ intersects both the interior and the complement of $\mathcal{N}$.

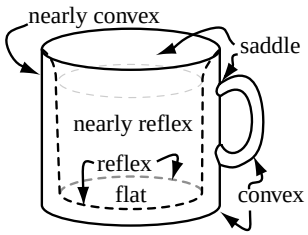


Figure 6.1: Classifying

For an example, we can classify points on the boundary of a three-manifold polyhedron. Points on faces are flat, points on edges are nearly reflex or nearly convex (or flat in the degenerate case of a dihedral angle of 180°), and points at vertices are convex, reflex, or saddle (except in degenerate cases of incident coplanar faces/edges). In a coffee mug, the reflex points are at the bottom of the bowl.

Given a closed half-space h^-, we call any bounded connected component of $\overline{\mathcal{Q}} \cap h^-$ a *plane-cavity*. A plane-cavity is maximal if no plane parallel to h defines a plane-cavity that contains it. It is not hard to see that you can fill a plane-cavity until it spills over at a saddle, in general, or at convex or nearly convex points in degenerate cases.

Lemma 23 *A polyhedral set $\mathcal{Q}$ has a plane-cavity $\overline{\mathcal{Q}} \cap h^-$ for some half-space h^- if and only if there is a reflex point in $\partial\mathcal{Q}$.*

Proof: Assume a point $p \in \partial\mathcal{Q}$ appears as a reflex point, which means that there is a neighborhood $\mathcal{N}$ of this appearance of p and a hemisphere such that $H_{v,\varepsilon}(p) \subset \text{int}(\mathcal{N})$. In the plane $h_v(p)$ that defines the hemisphere, choose a circle γ centered at p with radius less than ε. Because every point of γ is in the interior of $\mathcal{N}$, and $\mathcal{N}$ is compact, there is some $\delta > 0$ so that translating γ to $\gamma + \delta v$ remains strictly inside $\mathcal{N}$. The halfspace $h_v^-(p + \delta v)$ thus cuts off a bounded connected component from $\overline{\mathcal{Q}}$.

For the inverse, let X be a plane-cavity of $h^- \cap \overline{\mathcal{Q}}$ for some half-space h^-. Take a large sphere S that contains X strictly in its interior. Now, move S until S touches the boundary of X; by making S sufficiently large, the contact between S and X will be a single point $p \in \partial\mathcal{Q}$ that is not in the plane h. Let v be a vector from p towards the center of the sphere S. We may choose $\varepsilon > 0$ and $\mathcal{N}$ to be the neighborhood of the appearance of p on X, and form a hemisphere $H_{v,\varepsilon}(p) \subset \text{int}(\mathcal{N})$. Thus, p is a reflex point of $\mathcal{Q}$. □

We say that $\mathcal{Q}$ is a *reflex-free set* iff $\mathcal{Q}$ is a polyhedral set that has no reflex points. By the previous lemma, a polyhedral set has no plane-cavities if and only if it is reflex-free. The reflex-free sets are closed under intersection, provided they remain polyhedral.

Lemma 24 *Let $\{\mathcal{Q}_\alpha\}$ be a family of reflex-free sets whose intersection is polyhedral. The intersection $\bigcap \mathcal{Q}_\alpha$ is also reflex-free.*

Proof: Suppose that some point p is in a plane-cavity of $\bigcap \mathcal{Q}_\alpha$ defined by half-space h^-. By Lemma 23, it is sufficient to show that for some α, point p is also in a plane-cavity of set $\mathcal{Q}_\alpha$.

Notice that the plane-cavities of the intersection can be written as a union of individual plane cavities:

$$\overline{\left(\bigcap \mathcal{Q}_\alpha\right)} \cap h^- = \bigcup (\overline{\mathcal{Q}_\alpha} \cap h^-).$$

Since p is in a bounded connected component of the union, point p must be in a connected component of $h^- \cap \overline{\mathcal{Q}_\alpha}$, for some α. This component must be bounded, since the component in the union is bounded. □

We can use intersections to sculpt reflex-free sets. For example, drilling a hole through a reflex-free set keeps the set reflex-free.

Lemma 25 *Let $\mathcal{Q}$ be a reflex-free set and X be a polyhedral set. If no reflex point of X is in* $\text{int}(\mathcal{Q})$*, then the intersection $\mathcal{Q} \cap X$ is reflex free.*

Proof: Consider a point p on the boundary of $\mathcal{Q} \cap X$: either $p \in \partial\mathcal{Q}$ or $p \in \partial X \cap \text{int}(\mathcal{Q})$. In the first case, we know that no hemisphere $H(p) \subset \text{int}(\mathcal{Q})$, and in the second we know that no hemisphere $H(p) \subset \text{int}(X)$. Thus, there can be no hemisphere in their intersection. □

6.3 The reflex-free hull

Define $Rfh(\mathcal{Q})$, the *reflex-free hull* of a set $\mathcal{Q}$, as the intersection of all reflex-free sets that contain $\mathcal{Q}$. For example, the reflex-free hull of a torus is itself; the reflex-free hull of a coffee cup would fill the cup but preserve the handle. The reflex-free hull of a set of discrete points would be these points, because any union of balls around the points is reflex free. The motivation for defining the reflex-free hull is to find a structure that surrounds a set, but fills in depressions or docking sites. We show that the reflex-free hull is idempotent.

Theorem 12 *For a closed set $\mathcal{Q}$, the reflex-free hull $Rfh(\mathcal{Q})$ satisfies $Rfh(Rfh(\mathcal{Q})) = Rfh(\mathcal{Q})$.*

Proof: Since $\mathcal{Q} \subseteq Rfh(\mathcal{Q})$, the sets whose intersection defines $Rfh(Rfh(\mathcal{Q}))$ are a subset of those that define $Rfh(\mathcal{Q})$. Thus, it is clear that $Rfh(\mathcal{Q}) \subseteq Rfh(Rfh(\mathcal{Q}))$. We prove the reverse inclusion. By the definition of $Rfh(\mathcal{Q})$, if a point p is not in $Rfh(\mathcal{Q})$, then there is a reflex-free set R_p that includes $\mathcal{Q}$ and not p. Since R_p participates in the intersection defining $Rfh(\mathcal{Q})$, we also know that $Rfh(\mathcal{Q}) \subseteq R_p$. But then R_p also participates in the intersection defining $Rfh(Rfh(\mathcal{Q}))$. Thus, p is not in $Rfh(Rfh(\mathcal{Q}))$. □

We can use sculpting to prove the following technical lemma. Define the ε-*tube* for a line segment s as $\mathcal{C}_\varepsilon(s) = \{x \mid d(x,s) < \varepsilon\}$.

Lemma 26 *Suppose that Y is a reflex-free set and that Y contains a point q in the convex hull of a finite set of points outside of Y, namely $\{p_1, p_2, \ldots, p_k\} \subset \overline{Y}$. For some $\varepsilon > 0$, the set $Y \setminus \bigcup_{1 \le i \le k} \mathcal{C}_\varepsilon(p_i q)$ is reflex free.*

Proof: We may choose $\varepsilon > 0$ so that the balls $B_\varepsilon(p_i)$ do not intersect Y. The union of ε-tubes, $X = \bigcup_{1 \le i \le k} \mathcal{C}_\varepsilon(p_i q)$ is an open set; its complement $\overline{X}$ is a closed set that has reflex points only on ball boundaries $\partial B_\varepsilon(p_i)$. By Lemma 25, the intersection $Y \cap \overline{X}$ is reflex free. □

We use Lemma 26 to show that the reflex-free hull $Rfh(\mathcal{Q})$ inherits its convex, nearly convex, and saddle points from the underlying set $\mathcal{Q}$.

Lemma 27 *For a closed set $\mathcal{Q}$, every convex, nearly convex, or saddle point p of $Rfh(\mathcal{Q})$ is a point of $\mathcal{Q}$. In particular, convex points of $Rfh(\mathcal{Q})$ are convex points of $\mathcal{Q}$, nearly convex points of $Rfh(\mathcal{Q})$ are convex or nearly convex points of $\mathcal{Q}$, and saddle points of $Rfh(\mathcal{Q})$ are convex, nearly convex or saddle points of $\mathcal{Q}$.*

Proof: Because $\mathcal{Q} \subset Rfh(\mathcal{Q})$, the points inside $\mathcal{Q}$ are clearly interior points of $Rfh(\mathcal{Q})$.

Consider a point $p \in Rfh(\mathcal{Q})$ that is outside of $\mathcal{Q}$. We may choose $\varepsilon > 0$ such that the ball $B_\varepsilon(p)$ does not intersect $\mathcal{Q}$. The difference $Y = B_\varepsilon(p) \setminus \mathcal{Q}$ must be convex, since otherwise we could find two, three, or four points in Y whose convex hull contains a point $q \in Rfh(\mathcal{Q})$ and then apply Lemma 26 to obtain a smaller reflex-free set that still contains $\mathcal{Q}$. It is readily checked that for a convex, nearly convex, or saddle point p, the difference Y is not a convex set.

Therefore, a convex, nearly convex, or saddle point $p \in Rfh(Q)$ must come from $\partial\mathcal{Q}$. By checking the hemispheres of p with respect to $Rfh(\mathcal{Q})$ and $\mathcal{Q}$, we observe that a convex point of $Rfh(\mathcal{Q})$ is a convex point of $\mathcal{Q}$, a nearly convex point of $Rfh(\mathcal{Q})$ is a convex or nearly convex point of $\mathcal{Q}$, and a saddle point p of $Rfh(\mathcal{Q})$ is a convex, nearly convex, or saddle point of $\mathcal{Q}$. □

6.4 The reflex-free hull of a polyhedron

In this section, we consider the reflex-free hull of a polyhedron. We define the *size* of a polyhedron to be the number of vertices, edges, and faces on its boundary. We show that the reflex-free hull of a polyhedron is a polyhedron of the same asymptotic size. We find this surprising in light of the high complexity of other definitions of hulls that we sketch in the next section.

Theorem 13 *The reflex-free hull of a polyhedron of size n is a reflex-free polyhedron whose size is* $O(n)$.

We use $\mathcal{P}$ as our polyhedron and establish a sequence of lemmas before formally proving this theorem. We first wish to show that the boundary of $Rfh(\mathcal{P})$ consists of flat faces, straight segment edges, and point vertices. Since the convex, nearly convex, and saddle points of $Rfh(\mathcal{P})$ come from $\mathcal{P}$ by Lemma 27, our main task is to show in Lemma 28 that the nearly reflex points form line segments. Next we observe in Lemma 29 that each convex edge of $\mathcal{P}$ contributes at most two vertices to $Rfh(\mathcal{P})$. This allows us to establish that $Rfh(\mathcal{P})$ is a polyhedron in Lemma 30. Finally, we establish the theorem by relating the size to genus, and bounding the genus of $Rfh(\mathcal{P})$.

Lemma 28 *For a polyhedral set $\mathcal{P}$, let R be the set of nearly reflex points of $Rfh(\mathcal{P})$ that do not lie at vertices or on edges of $\mathcal{P}$. The connected subsets of R are line segments.*

Proof: Let $p \in R$ be a nearly reflex point of $Rfh(\mathcal{P})$ that is not a vertex of $\mathcal{P}$. Choose $\varepsilon > 0$ sufficiently small that the only facets of $\mathcal{P}$ that intersect $B_\varepsilon(p)$ are those incident on p. Let h^- be the halfspace that contains the hemisphere that shows that (this appearance of) p is nearly reflex.

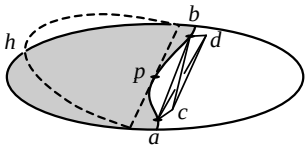


Figure 6.2: Sculpt with tetrahedron *abcd* or rotate *h*.

Figure 6.2 illustrates the disk $D = h \cap B_\varepsilon(p)$ that is contained in $Rfh(\mathcal{P})$, drawn with shading where D intersects the interior $\text{int}(Rfh(\mathcal{P}))$. Notice that the points $R \cap D$ serve as the boundary between shaded and unshaded, that is, between $D \cap \text{int}(Rfh(\mathcal{P}))$ and $D \cap \partial Rfh(\mathcal{P})$.

Choose a and b as points of $R \cap B_\varepsilon(p)$, as in Figure 6.2. We observe that the segment ab cannot intersect the interior $\text{int}(Rfh(\mathcal{P}))$: Construct an open tetrahedron τ as the interior of the convex hull of a, b, and two other points, $c, d \in B_\varepsilon(p) \setminus Rfh(\mathcal{P})$ so that a, b, c, and d are not coplanar. The reflex vertices of $\overline{\tau}$ are a, b, c, and d, which are not in the interior of $Rfh(\mathcal{P})$. The open tetrahedron τ does not intersect $\mathcal{P}$, since c and d lie outside and a and b lie on or outside the one facet of $\mathcal{P}$ in $B_\varepsilon(p)$. Thus, $Rfh(\mathcal{P}) \cap \overline{\tau}$ is reflex free by Lemma 25. But this sculpting operation cannot remove points from $Rfh(\mathcal{P})$, so the segment ab does not intersect the interior $\text{int}(Rfh(\mathcal{P}))$.

Now, consider the convex hull of the nearly reflex points R within disk D. This hull cannot contain a point of the interior, $D \cap \text{int}(Rfh(\mathcal{P}))$, by the previous paragraph. Since p is not flat, there are interior points in any neighborhood of p, so p must be on the boundary of the hull. If we rotate the plane h around a tangent to the hull at p and define a new hemisphere $H(p)$ as indicated by the dashed line in Figure 6.2, then we can see that p must lie on a line segment on this hull, or we would observe a reflex point at p. This concludes the proof of the lemma. □

We next observe that an edge from $\mathcal{P}$ appears at most once as a nearly convex edge on $Rfh(\mathcal{P})$.

Lemma 29 *If p and q are nearly convex points of Rfh($\mathcal{P}$) that come from the same edge e of $\mathcal{P}$, then the segment pq consists entirely of nearly convex points of Rfh($\mathcal{P}$).*

Proof: Let p and q be points satisfying the hypothesis of the lemma. Choose $\varepsilon > 0$ sufficiently small that the ε-tube $\mathcal{C}_\varepsilon$(pq) does not intersect any facets of $\mathcal{P}$ except those incident on segment pq. At the left side of Figure 6.3, we draw two ε disks around p and q, and shade their intersections with $\mathcal{P}$ (dark) and $Rfh(\mathcal{P})$ (light).

We may choose points a and b outside of $Rfh(\mathcal{P})$ so that the segment ab is parallel to pq: simply choose a and b in the antipodes of the regions intersecting $\mathcal{P}$ in the two disks, as in the middle of Figure 6.3. We may then choose c and d outside of $Rfh(\mathcal{P})$ so that $\angle apc$

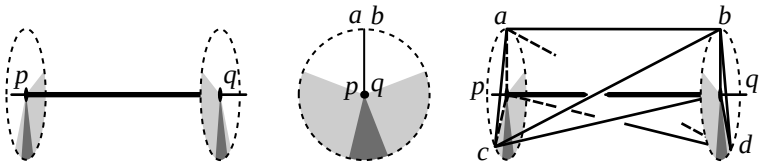


Figure 6.3: Sculpting $Rfh(\mathcal{P})$ between two nearly convex points p and q of an edge e of $\mathcal{P}$.

is obtuse in the disk containing p and $\angle bqd$ is obtuse in the disk containing q, and c and d lie on opposite sides of the plane through $abpq$.

Now, form the sculpting region X as the closure of the complement of the two tetrahedra $acpq$ and $bdpq$, as at the right of Figure 6.3. The reflex points of X, which are a, b, c, and d, are all outside of $Rfh(\mathcal{P})$, so by Lemma 25 we know that $Rfh(\mathcal{P}) \subset X$. Since all points of pq are on $e \subset Rfh(\mathcal{P})$, we can find hemispheres to show that they are all nearly convex points of $Rfh(\mathcal{P})$. □

We summarize what we know about $Rfh(\mathcal{P})$ thus far.

Lemma 30 *The reflex-free hull of a polyhedron of size n is bounded by a polyhedron with $O(n)$ vertices.*

Proof: Let $\mathcal{P}$ be a polyhedron of size n, and classify the points of $\partial Rfh(\mathcal{P})$. By Lemmas 27 and 28, the nearly convex and nearly reflex points may be organized into line segments. These line segments must end at vertices that are convex or saddle points, since there are no reflex points. But each such vertex lies on a nearly convex edge of $\mathcal{P}$, and each edge can contribute at most two vertices by Lemma 29. The remaining points are flat points, which may therefore be grouped into polygons. A finite number of polygons may be formed on n vertices, so $Rfh(\mathcal{P})$ is a polyhedron. □

We can now complete the proof of Theorem 13, and show that, for a polyhedron $\mathcal{P}$ the reflex-free hull $Rfh(\mathcal{P})$ is a polyhedron of the same asymptotic size.

Proof: From Lemma 30 we know that $Rfh(\mathcal{P})$ is a polyhedron with $O(n)$ vertices, where n is the size of $\mathcal{P}$. We have only to bound the numbers of faces and edges.

From the Euler-Poincaré formula (due to Poincaré 1899), we know that $V - E + F = 2 - 2g$, where V, E, F, and g are the number of vertices, edges, faces, and genus of $Rfh(\mathcal{P})$. Since $3F \leq 2E$, we deduce that $V - E/3 \geq 2 - 2g$ and so $E \leq 3V + 6g - 6$. Since $V \in O(n)$, we have only to bound the genus $g \in O(n)$ to complete the proof.

Curvature is defined at all points on a surface, and the sum of curvature over $Rfh(\mathcal{P})$ equals $-4\pi(g-1)$. For a polyhedron, points on faces or edges have curvature zero, and the curvature of a vertex v equals 2π minus the sum of the angles of faces incident on v.

Thus, $4\pi g$ equals 2π plus the sum of all face angles at all vertices of $Rfh(\mathcal{P})$. Since the genus of $\mathcal{P}$ is less than n by the Euler-Poincare formula, we bound the increase of genus by bounding the increase in the sum of face angles when we go from $\mathcal{P}$ to $Rfh(\mathcal{P})$.

Three types of changes to vertices occur when we go from $\mathcal{P}$ to $Rfh(\mathcal{P})$:

1. a vertex v of $\mathcal{P}$ may disappear into the interior of $Rfh(\mathcal{P})$,
2. a new vertex v may be created on $Rfh(\mathcal{P})$, or
3. a vertex v of $\mathcal{P}$ may become incident on new faces.

We can bound how each type of change increases the sum of face angles.

1. When a vertex v of $\mathcal{P}$ disappears, it no longer contributes to the sum of face angles.
2. A new vertex v is incident on exactly one convex edge, since v is not reflex and is not a vertex in $\mathcal{P}$. The sum of angles of the two faces incident to the convex edge is less than 2π, and the sum of angles of the remaining faces incident to v is less than 2π. So a new vertex increases the sum of face angles by less than 4π.
3. Since $Rfh(\mathcal{P})$ is a polyhedron, we may organize the faces incident to v into one or more topological disks. At most one of these disks may be flat, contributing an angle of 2π. Any other must have at least one convex edge.

   Lemma 27 implies that no new convex edge can be created incident to v. Thus, this change replaces the faces between two convex edges (possibly identical) by a new set of faces that are joined by reflex edges. This decreases the sum of face angles, except possibly where a new face angle of greater than π is created incident to v. This new face adds less than 2π, but also consumes one quarter of the neighborhood of v. The increase in face angles at v will be less than 8π.

Hence, the maximum increase in face angles is $8\pi n$, and the increase in genus of $Rfh(\mathcal{P})$ is thus $O(n)$. This completes the proof of Theorem 13. □

6.5 The reflex-free hull and cavities

Recall that for a closed polyhedral set $\mathcal{Q}$ and halfspace h^-, we defined a *plane-cavity* as a connected component of $\overline{\mathcal{Q}} \cap h^-$. We can enlarge a plane-cavity by translating its plane h unless h contains a saddle point or nearly convex points. We say that a plane-cavity is *limited* if its plane contains three saddle points or a closed curve of convex and nearly convex points.

Lemma 31 *Any plane-cavity is contained in the union of four limited plane-cavities.*

Proof: Consider a plane-cavity $C \subset \overline{\mathcal{Q}} \cap h^-$. We may translate h to enlarge C unless doing so would cause C to be connected to the unbounded component of $\overline{\mathcal{Q}}$. This may happen if h contains a closed chain of (nearly) convex points satisfying the lemma.

Otherwise h contains one or more saddle points. If h contains two saddle points, a and b, then we may duplicate h and rotate the two copies in opposite directions around the line ab. We stop each rotation when a third saddle point or chain of nearly convex points is reached. If there is a single saddle, we again duplicate and rotate h around some line through the saddle until we hit a second saddle and reduce to two instances of the previous case. □

If we iteratively fill up plane cavities for a polyhedron $\mathcal{P}$, then we obtain a sequence of interesting sets. We describe this process precisely as follows. Let $\mathcal{P}_0 = \mathcal{P}$. Given some plane-cavity C_k of $\mathcal{P}_k$, we form the union $\mathcal{P}_k \cup C_k$ to obtain a new polyhedron $\mathcal{P}_{k+1}$. We may choose our plane-cavities by always choosing the one with largest volume, or by always choosing four limited plane-cavities that enclose the largest volume.

We call a connected component of $\mathcal{P}_k \setminus \mathcal{P}$ a *cavity*. We believe, but have not been able to formally prove, that in the limit we can obtain the reflex-free hull by filling cavities. Equivalently the cavities of a closed polyhedron $\mathcal{P}$ are the connected components of $Rfh(\mathcal{P}) \setminus \mathcal{P}$.

Theorem 14 *For a closed polyhedron $\mathcal{P}$, the limit of the process of filling cavities is a subset of the reflex-free hull, Rfh($\mathcal{P}$).*

Proof: We show by induction that the cavities identified by the filling process are inside all reflex-free sets that contain $\mathcal{P}$. Specifically, we prove that any polyhedral set $\mathcal{Q}$ that contains $\mathcal{P}$, but does not contain $\mathcal{P}_k$, has a reflex point.

The base case, $k = 0$, is trivial; no set containing $\mathcal{P}$ can omit a point of $\mathcal{P}_0 = \mathcal{P}$.

For the inductive step, we assume the induction hypothesis for some $k \geq 0$, and prove it for $k+1$. Thus, assume that $\mathcal{Q}$ is a polyhedral set that contains $\mathcal{P}$ but does not contain $\mathcal{P}_{k+1}$. If $\mathcal{Q}$ does not contain $\mathcal{P}_k$, then the induction hypothesis applies immediately, so we assume that $\mathcal{Q}$ contains $\mathcal{P}_k$. The boundary of $\mathcal{Q}$ therefore intersects the plane-cavity $C_k = \mathcal{P}_{k+1} \setminus \mathcal{P}_k$. But since $\partial\mathcal{Q}$ can only escape through the plane defining C_k, if at all, $\mathcal{Q}$ has a plane cavity in C_k. Lemma 23 says that $\mathcal{Q}$ has a reflex point. □

Unfortunately, we do not know how to turn this definition into an efficient procedure to compute the reflex-free hull of a polyhedron. The process of filling in one reflex vertex can create others at reflex edges. Figure 6.4 illustrates one example in which filling cavities must be taken to the limit to attain the reflex-free hull.

Start with the cube $[-5,5]^3$ and subtract the following sets: $\{(x,y,z) \mid x \in [-1,1], z > |y|\}$, $\{(x,y,z) \mid |x| \in [1,3], z > |y| - |x| + 1\}$, $\{(x,y,z) \mid x \in [3,5], z > y/2 + 1\}$, and $\{(x,y,z) \mid x \in [-3,-5], z > -y/2 + 1\}$, to obtain an object illustrated in Figure 6.4. There are four

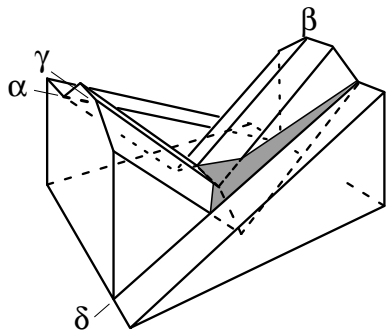


Figure 6.4: Each filling step creates reflex and saddle points

labeled lines that are relevant in this example. We parameterize them by z. Two are *pivots*, $\alpha = \{(-3, 2z-2, z)\}$ and $\delta = \{(3, 2-2z, z)\}$, and two lines end at saddle points, $\beta = \{(-1, z, z)\}$, and $\gamma = \{(1, -z, z)\}$. With a little algebra, we observe that a plane that contains δ and intersects β at $z = t$ intersects γ at $z = (t+2)/6$. By symmetry, the plane through α and $(1, -t, t)$ intersects β at $z = (t+2)/6$.

Initially, there are two reflex vertices with coordinates $(\pm 3, 0, -3)$. We can eliminate the first by filling the cavity defined by the plane through pivot δ and the saddle point at $z_0 = 0$ on β; this plane intersects γ at $z_1 = 2/6$. We eliminate the second by filling to the plane through α and saddle point $(1, z_1, z_1)$; this plane intersects β at $z_2 = 7/18$, and creates a new reflex vertex where the two filling planes meet. From now on, we fill from a pivot line to a saddle at $z_i = (z_{i-1} + 2)/6$. The reflex-free hull for this example has a reflex edge along the line through $(-1, 2/5, 2/5)$ and $(1, -2/5, 2/5)$, which happens to be the unique line incident to α, β, γ, and δ. Thus, we approach, but never reach the reflex-free hull.

If reflex edges incident on four polyhedron edges were the worst that could occur, we could still hope for a polynomial-time algorithm for the reflex-free hull by inspecting all 4-tuples of edges to see if they support a common line. Unfortunately, however, reflex edges may be defined by lines that hit only two polyhedron edges. Figure 6.5 shows such an example made of eight spheres, which could be approximated by polyhedra. In it, we have a sequence of eight geodesic triangles that share the thick segments, which are reflex edges of the hull. The extensions of reflex edges (dash-dotted) intersect within the incident geodesic triangles. The reader who would like to find an algorithm to compute reflex-free hulls is advised to build similar examples from modeling clay.

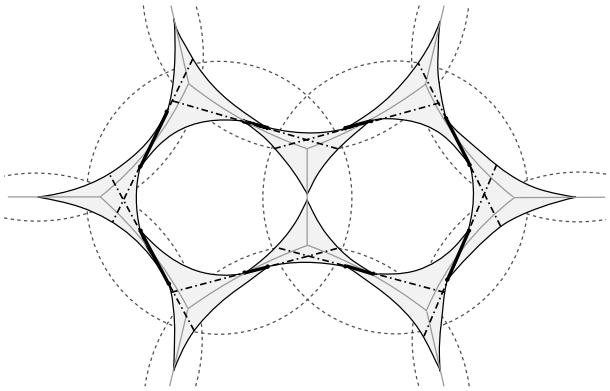


Figure 6.5: Part of the reflex hull of an appropriately placed set of eight spheres. The planes determining the new boundaries of this hull are defined by a sequence of saddles in such a way that if any saddle is moved then all plane equations change.

6.6 Other hulls

The fact that the reflex-free hull has linear complexity may not at first seem surprising. In this section, we consider some other natural definitions for hulls that have far worse complexities.

For a closed set S, we may obtain the convex hull, $CH(S)$, by removing halfspaces that do not intersect S or by taking the intersection of halfspaces that contains S. We may obtain the reflex-free hull, $Rfh(S)$, by sculpting according to Lemma 25, or by taking the intersection of reflex-free sets that contian S. In a similar manner, we can define a line hull, $LH(S)$, by removing lines that do not intersect S, or more formally by taking the intersection of sets containing S that are the complements of lines.

Lemma 32 *For a polyhedral set S, we have $S \subseteq Rfh(S) \subseteq LH(S) \subseteq CH(S)$. In general, all inclusions are strict.*

Proof: It follows immediately from the definitions. The complement of a line is reflex-free, and a halfplane can be represented as the intersection of the complements of lines. □

In the plane, the line hull of a connected set is the same as its convex hull, but the line hull of a disconnected polyhedral set of size n may have $\Theta(n^4)$ complexity, as it is related to an arrangement of the $\Theta(n^2)$ lines tangent to pairs of vertices of S (See Figure 6.6.) In $\mathbb{R}^3$, the line hull is bounded by pieces of ruled surfaces, including hyperboloids. Sergei Bespamyatnikh, in private communication, described an example of a connected polyhedral set S of size n whose line hull has $\Theta(n^9)$ complexity. Begin with the six faces of a

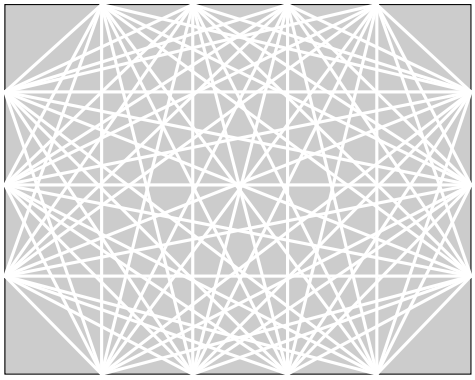


Figure 6.6: Line hull of 18 black segments

large, axis-aligned cube, and cut a small square hole in the center of each face. Near the center of this cube we have three families of lines, each roughly parallel to one of the three axes. Block the lines parallel to the x-axis with three squares parallel to the yz plane, and cut n parallel slits in different directions each square so that the lines that do pass through these slits form $\Omega(n^3)$ hyperboloids near the center of the cube. Repeat this for the y- and z-axes, so that these hyperboloids have $\Omega(n^9)$ intersections.

Some other natural hull definitions suffer from similar complexities. One could define the *star hull*, $SH(S)$, of S as the intersection of all star-shaped polyhedra that contain S. Each point p that is not in the star hull is excluded from some star-shaped polyhedron, which says that p has a ray to infinity that does not intersect the interior of S. Thus, one could define the *ray hull* as the intersection of sets containing S that are the complements of open rays. We say that a point p of a closed set X is *externally visible* if there is a ray from p that does not intersect the interior of X. A set X is *externally visible* if every point on its boundary is externally visible. Thus, one could define the *externally visible hull* of a closed set S to be the intersection of all externally visible sets that contain S. It is not difficult to see that the star hull, ray hull, and externally visible hull are identical, and that $S \subseteq SH(S) \subseteq LH(S)$, with strict inclusion for many sets S. The reflex-free hull and star hull cannot always be ordered by inclusion: The boundary of the star hull contains any reflex vertices from S that are externally visible. The boundary of the reflex-free hull may contain points that are not externally visible, as can be seen in an example of nested tori rotated about a common axis so they form a spherical shell.

A minor modification of Bespamyatnikh's construction shows that the star hull can again be bounded by hyperboloids, and may have $\Omega(n^9)$ complexity.

CHAPTER 7

Cavities and castability analysis

7.1 Introduction

Feature recognition has been considered an important research area in computer-aided design and computer-aided manufacturing [43, 44, 24]. Informally, features are product's generic shapes or characteristics that are associated with properties, attributes, and engineering knowledge about the product [46, 47]. Manufacturing features are geometric structures of an object, such as holes or depressions, which have engineering meanings related to manufacturing operations. A hole of an object, for example, may have an engineering meaning "drilling" or "assembling site".

In applications such as manufacturing and molecular analysis, geometric structures such as cavities or docking sites are important. In manufacturing, features of a CAD model imply manufacturing information, which facilitates the process of analyzing manufacturability [44, 23].

A small hole or a depression on the boundary of an object, for example, restricts the set of directions for which this object is castable, because the portion of the cast in the hole or in the depression must be removed from the object without breaking the object. Most industrial parts such as engine rooms, telephone bodies, and small parts for car and aircraft have such features. This suggests a new approach to castability analysis: For given part removal directions, instead of examining the whole boundary of an object, we identify such features (holes and depressions) which play key roles in the preliminary decision process. If any such features contradicts the removal directions, we can stop and conclude that these directions are not feasible, or that the object needs additional cast

parts. So identifying features not only facilitates the decision process and the automated design of a cast, but also greatly reduces the search space for feasible casting directions. When we search for the set of all feasible casting directions, features can greatly reduce the search space. A hole with the shape of cylinder in an object, for example, reduces the search space to a pair of two opposite directions parallel to the generator of the cylinder. Features, furthermore, can be used to minimize the number of casting parts (called *side cores*).

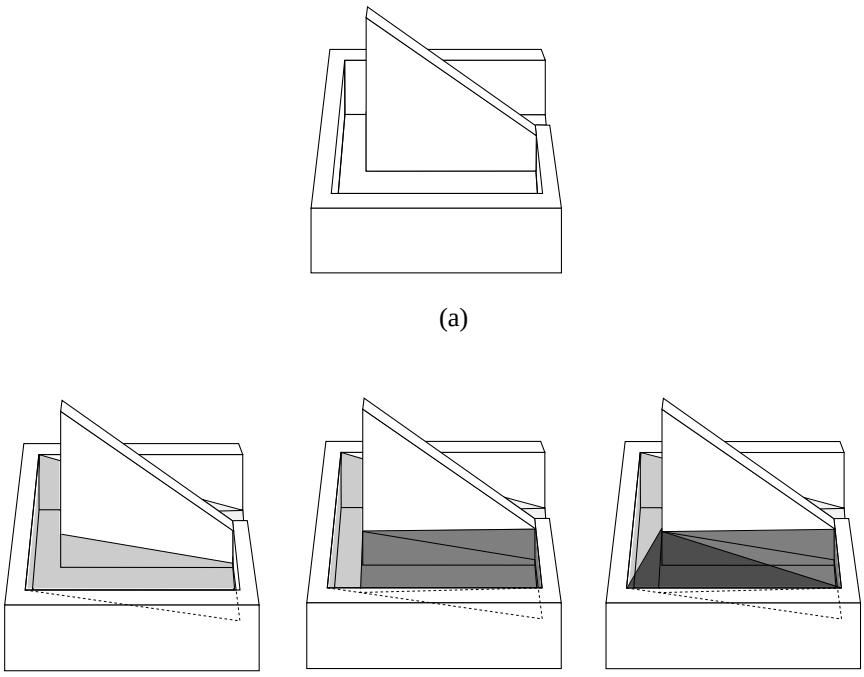
Based on the definitions of the reflex-free hull and cavities in Chapter 6, in this chapter we consider applications using cavities as a geometric feature in castability analysis. We assume that the cast (mould) consists of two parts and that these parts must be removed in opposite direction without damaging the parts or the object.

We present an algorithm which is useful for casting analysis. The algorithm partitions the faces of $\mathcal{P}$ into disjoint subsets, such that each subset must belong to one of the two mould parts. Furthermore, we prove that the bounding faces of a cavity belong to a single subset. By basing the algorithm on faces, we obtain a finite process. Our algorithm is an effective method to restrict the search space for feasible casting directions. In fact, we conjecture that this algorithm can be extended so that, in the end, for any two distinct subsets, there is a feasible casting direction in which the mould is removed from the corresponding faces in opposite directions.

7.2 Definitions and assumptions

Recall the process of iteratively filling plane-cavities of Section 6.5. We denote this process by $(\mathcal{P}, \sigma_k)$, where σ_k is any sequence of k plane-cavities. A connected component of $\mathcal{P}_k \setminus \mathcal{P}$ is a *cavity* of $(\mathcal{P}, \sigma_k)$. A face f of $\mathcal{P}$ *bounds* a cavity $\mathcal{F}$ of $(\mathcal{P}, \sigma_k)$ if f lies partially or completely on the boundary of $\mathcal{F}$. For the rest, we follow the definitions and the notations in Section 6.2.

We make one assumption for robustness: removal directions parallel to faces of $\mathcal{P}$ are not allowed. Thus, when the two parts of a mould for $\mathcal{P}$ meet along a *parting surface*, this parting surface meets the boundary of $\mathcal{P}$ along a closed curve called the *parting line* that consists of polyhedron edges. This is a practical consideration in mould design as well, since casting imperfections on the object may occur along the parting line, and if the parting line crosses a face then additional treatment and polishing may be required. For further information, see Section 4.1.



(a)

(b)

Figure 7.1: Cavities. (a) A container object, and (b) Filling process.

7.3 Theorems for coloring faces

In this section we show how to color the faces of $\mathcal{P}$ such that faces with the same color must appear in the same part of a two-part mould.

This algorithm is based on two geometric observations: Every polyhedron face belongs to one of the two mould parts of any cast, and the two polyhedron faces incident to a reflex edge must belong to the same mould part of any cast.

Lemma 33 *The bounding faces of a cavity of $(\mathcal{P},\sigma_k)$ for any σ_k must belong to the same mould part of any cast.*

Proof: Given a cavity $\mathcal{F}$ of $(\mathcal{P},\sigma_k)$, the boundary of $\mathrm{cl}(\mathcal{F})$ can be divided into two parts: one shared with $\mathcal{P}$ and one not shared with $\mathcal{P}$. We call the part not shared with $\mathcal{P}$ the *lid* of $\mathcal{F}$. It suffices to prove that no two points at the boundary between $\mathcal{P}$ and a cavity can

be removed in opposite directions.

Assume that two distinct points on the bounding faces of $\mathcal{F}$ are removed in opposite directions. Inside the cavity the two cast parts meet along a common boundary surface. Then any line ℓ through the interior of the surface and parallel to the removal direction does not intersect $\mathcal{P}$. Since the reflex-free hull of $\mathcal{P}$ does not contain the intersection point between ℓ and the surface, $\mathcal{P}_k$ is not a subset of the reflex-free hull of $\mathcal{P}$, which contradicts Theorem 14. □

We need a few definitions in order to describe our algorithm. Let $\mathcal{S}$ be the sphere of directions. Given a face f, we denote by $cone(f)$ the set of directions on $\mathcal{S}$ that have positive projection on the normal of f. Note that if we translate f such that f passes through the center of $\mathcal{S}$, then $cone(f)$ is an open hemisphere defined by a plane through f. Symmetrically, we define $\overline{cone}(f)$ to be the set of directions on $\mathcal{S}$ that have negative projection on the normal of f. We denote the double cone $cone(f) \cup \overline{cone}(f)$ by $dcone(f)$. We generalize the notation to reflex edges. Given a reflex edge e with incident faces f and g, define $cone(e) = cone(f) \cap cone(g)$, $\overline{cone}(e) = \overline{cone}(f) \cap \overline{cone}(g)$, and $dcone(e) = cone(e) \cup \overline{cone}(e)$. Note that $cone(e)$ is the set of removal directions for f and g. (Recall that f and g must belong to the same mould part by assumption.)

Our algorithm works by assigning a color and a positive or negative sign to each face. Faces of the same color (regardless of the sign) form a *color group*. Given a color group G, we define its *cone of directions*, $cone(G)$, to be the common intersection of $cone(f)$ for all positive faces $f \in G$ and $\overline{cone}(g)$ for all negative faces $g \in G$. Symmetrically, $\overline{cone}(G)$ is the set of directions opposite to those in $cone(G)$. The double cone $dcone(G)$ is $cone(G) \cup \overline{cone}(G)$.

The algorithm consists of two phases. Initially, each face is assigned a positive sign and a distinct color, and therefore forms a color group by itself. In the first phase, we repeatedly recolor two groups G_1 and G_2 of faces by one common color if G_1 and G_2 meet along some reflex edge. In the second phase, we repeatedly recolor two groups G_1 and G_2 of faces by one common color if $dcone(G_1) \cap dcone(G_2)$ consists of exactly two connected components. We may also update the signs of faces in $G_1 \cup G_2$ and there are two cases: (1) $cone(G_1) \cap cone(G_2) \neq \emptyset$ and $cone(G_1) \cap \overline{cone}(G_2) = \emptyset$, (2) $cone(G_1) \cap cone(G_2) = \emptyset$ and $cone(G_1) \cap \overline{cone}(G_2) \neq \emptyset$. In case (1), we preserve the signs of all faces in $G_1 \cup G_2$. In case (2), we flip the sign of each face in G_2.

Lemma 34 *For any color group G, all positive faces in G must be removed in a common direction in $cone(G)$, and all negative faces in G in a common direction in $\overline{cone}(G)$, with respect to any mould.*

Proof: We prove this by induction. In the first phase, two faces f and g incident to a reflex edge must be removed in the same direction by our assumption that no face is parallel to a casting direction. In the second phase, suppose we decide to combine two color groups

G_1 and G_2. By induction assumption, all positive (resp. negative) faces in G_1 must be removed in a common direction in $cone(G_1)$ (resp. $\overline{cone}(G_1)$) with respect to any cast, and the same holds for G_2.

Suppose that $cone(G_1) \cap \overline{cone}(G_2) = \emptyset$. Then positive faces in G_1 cannot be removed in a direction in $\overline{cone}(G_2)$ and vice versa. Thus, positive faces in $G_1 \cup G_2$ must be removed in a common direction with respect to any cast, and this set of common directions is clearly $cone(G_1) \cap cone(G_2)$. A symmetric statement holds for negative faces in $G_1 \cup G_2$ and $\overline{cone}(G_1) \cap \overline{cone}(G_2)$.

Suppose that $cone(G_1) \cap cone(G_2) = \emptyset$. Then positive faces in G_1 cannot be removed in a direction in $cone(G_2)$ and vice versa. Thus, positive faces in G_1 and negative faces in G_2 must be removed in a common direction in $cone(G_1) \cap \overline{cone}(G_2)$. Since signs of faces in G_2 are flipped in merging, the lemma is satisfied. A symmetric statement holds for negative faces in G_1 and positive faces in G_2. □

Lemma 35 *Let $\mathcal{P}$ be a castable polyhedron. Let f and g be two bounding faces of a cavity of $(\mathcal{P}, \sigma_k)$ for some σ_k. Suppose that f and g belong to two different color groups G_1 and G_2 at some point during the coloring algorithm. If f and g have identical signs, then $cone(G_1) \cap cone(G_2)$ is nonempty. Otherwise, $cone(G_1) \cap \overline{cone}(G_2)$ is nonempty.*

Proof: Let $\mathcal{C}$ be a two-part mould for $\mathcal{P}$. By Lemma 33, f and g belong to the same part of $\mathcal{C}$. If f and g have identical sign, then Lemma 34 implies that faces in $G_1 \cup G_2$ of the same sign belong to the same part of $\mathcal{C}$. Thus, the removal direction of the positive faces in $G_1 \cup G_2$ belongs to $cone(G_1) \cap cone(G_2)$ which must then be nonempty. If f and g have opposite signs, then Lemma 34 implies that positive (resp. negative) faces in G_1 and negative (resp. positive) faces in G_2 belong to the same part of $\mathcal{C}$. Thus, the removal direction of positive faces in G_1 and negative faces in G_2 belongs to $cone(G_1) \cap \overline{cone}(G_2)$ which must then nonempty. □

We will prove that the bounding faces of a cavity of $(\mathcal{P}, \sigma_k)$ for any k will receive the same color. The proof is by induction on k. Since a face f will reside in different color groups G during the coloring algorithm, the set of removal directions for f changes as G changes. For ease of exposition, we disregard the group that f belongs to. Instead, we say that different cones of directions D are *associated with* f at different times during the coloring algorithm.

Furthermore, in making the inductive argument, we need to work with plane-cavity with respect to a nearly reflex vertex instead of a reflex vertex. Recall that v is a nearly reflex vertex if v is neither reflex nor flat, and all faces incident to v lie within a closed halfspace whose bounding plane passes through v and lies locally inside $\mathcal{P}$ at v.

Define the *star* of a vertex v, $St(v)$, to be the union of v and the interior of faces and edges incident to v. We use $\overline{St}(v)$ to denote the closure of $St(v)$. The *link* of v, denoted by $Lk(v)$,

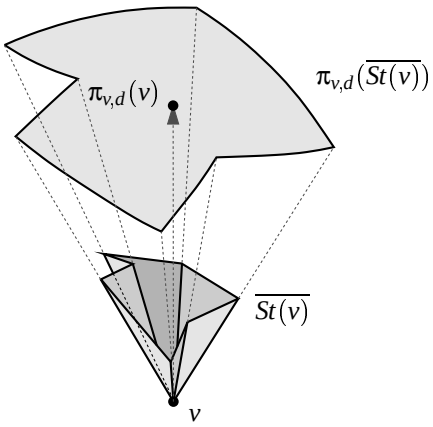


Figure 7.2: The closure, $\overline{St}(v)$, of the star of a reflex vertex v and the spherical polygon $\pi_{v,d}(\overline{St}(v))$ on the sphere, where d is the upward vertial direction.

is defined to be $\overline{St}(v) \setminus St(v)$. Let v be a reflex or nearly reflex vertex. Let d be any feasible casting direction from which v is visible. Note that all faces incident on v are visible from direction d. Put v at the center of the sphere of direction. First, we stretch $\overline{St}(v)$ radially away from v so that the stretched link of v lies on the sphere. This yields a polygon with curved boundary inside the sphere. Second, we project v in direction d onto the sphere. We use $\pi_{v,d}$ to denote the composite mapping from $\overline{St}(v)$ to the spherical polygon on the sphere. Note that $\pi_{v,d}(v)$ is in the kernel of $\pi_{v,d}(\overline{St}(v))$. Thus, $\pi_{v,d}(\overline{St}(v))$ can be triangulated into spherical triangles by drawing great circular arcs from $\pi_{v,d}(v)$ to vertices of $\pi_{v,d}(\overline{St}(v))$. Clearly, $\pi_{v,d}$ maps the circular arcs incident to $\pi_{v,d}(v)$ to edges incident to v, and the spherical triangles to triangles. Also, the angle at a vertex $\pi_{v,d}(x)$ of $\pi_{v,d}(\overline{St}(v))$ is exactly the exterior dihedral angle at the edge vx.

Lemma 36 *Let v be a reflex or nearly reflex vertex. Let va and vb be two reflex edges incident to v. Let $D_{va} \subseteq cone(va)$ and $D_{vb} \subseteq cone(vb)$ be two cones of directions associated with the two faces incident to va and vb, respectively. If $cone(va)$ and $cone(vb)$ lie on the same side of a great circle through $\pi_{v,d}(a)$ and $\pi_{v,d}(b)$, then $D_{va} \cap \overline{D_{vb}}$ is empty.*

Proof: Figure 7.3 illustrates the situation. It follows from the fact that $cone(vx)$ and $\overline{cone}(vx)$ for any reflex edge vx lie on opposite sides of any great circle that does not intersect $cone(vx)$ (such a great circle must pass through $\pi_{v,d}(x)$). □

Lemma 37 *Let v be a reflex or nearly reflex vertex. Let d be a feasible removal direction from which v is visible. Let va, vb, and vx be three reflex edges such that $\pi_{v,d}(vx)$ lies in the smaller angle between $\pi_{v,d}(va)$ and $\pi_{v,d}(vb)$. Suppose that $cone(v\alpha)$ does not contain*

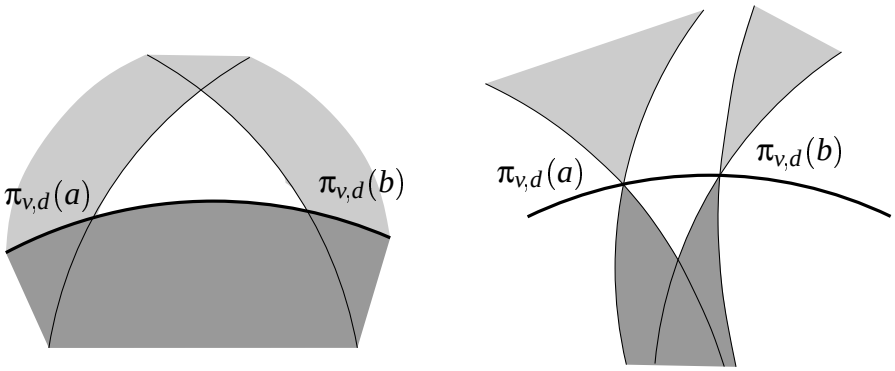


Figure 7.3: The bold curves are parts of great circles. The left picture shows the case where *va* and *vb* are incident to the same face. The right picture shows the case where they are not. The area of darker shade represents $cone(va)$ and $cone(vb)$. The area of lighter shade represents $\overline{cone}(va)$ and $\overline{cone}(vb)$. Since $cone(va)$ and $\overline{cone}(vb)$ lie opposite sides of the great circle, they cannot intersect. The same is true for $\overline{cone}(va)$ and $cone(vb)$.

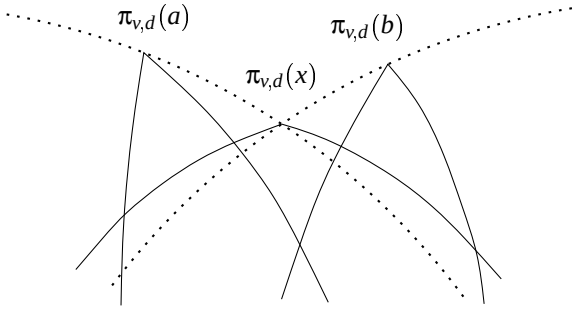


Figure 7.4: The dotted curves are great circles through $\pi_{v,d}(a)$ and $\pi_{v,d}(x)$, and $\pi_{v,d}(b)$ and $\pi_{v,d}(x)$.

$\pi_{v,d}(\beta)$ *for all* $\alpha, \beta \in \{a, b, x\}$. *Then* $D \cap \overline{D_x}$ *is empty, where* $D \subseteq cone(va) \cap cone(vb)$ *is a common cone of directions associated with the faces incident to va and vb, and* D_x *is the cone of directions associated with the faces incident to vx.*

Proof: Since $\pi_{v,d}(v)$ lies in the kernel of $\pi_{v,d}(St(v))$, $cone(vx)$ and $cone(va)$ cannot lie on opposite sides of the great circle through $\pi_{v,d}(a)$ and $\pi_{v,d}(x)$. The same holds for $cone(vx)$ and $cone(vb)$. If $cone(vx)$ and $cone(va)$ lies on the same side of the great circle through $\pi_{v,d}(a)$ and $\pi_{v,d}(x)$, then Lemma 36 implies that $D \cap \overline{D_x}$ is empty. We obtain the same conclusion if $cone(vx)$ and $cone(vb)$ lies on the same side of the great circle through $\pi_{v,d}(b)$ and $\pi_{v,d}(x)$. The remaining possibility is that $cone(vx)$ contains $cone(va) \cap cone(vb)$

which is a superset of D. See Figure 7.4. Thus, D cannot intersect $\overline{cone}(vx)$ which is a superset of $\overline{D_x}$. □

We are ready to prove that the bounding faces of a cavity of $(\mathcal{P}, \sigma_k)$ for any σ_k receive the same color and sign. We will restrict σ_k to simplify the analysis without loss of generality. Specifically, we want to refine σ_k to a sequence of *special* plane-cavities such that the filling of each special plane-cavity corresponds to sweeping a plane from a reflex or nearly reflex vertex until a vertex in the cavity is encountered. Given σ_k, we can refine it as follows: When we fill the plane-cavity $\sigma_k \setminus \sigma_{k-1}$ of $\mathcal{P}_{k-1}$ to produce $\mathcal{P}_k$, the plane-cavity can be decomposed into several steps. Take the plane H_k defining the plane-cavity $\sigma_k \setminus \sigma_{k-1}$. Sweep H_k towards the interior of $\sigma_k \setminus \sigma_{k-1}$ until it hits a vertex w of the plane-cavity. Record the volume swept over as one special plane-cavity. Continue the sweeping to the next vertex in the plane cavity and define another special plane-cavity. During the sweeping, we may need to split at the vertex encountered. In this case, we continue the sweeping of the different parts independently. Figure 7.5 shows an example. We repeat the above sweeping until no vertex in $\sigma_k \setminus \sigma_{k-1}$ remains. Now, we can think of the growing of $\mathcal{P}_{k-1}$ to $\mathcal{P}_k$ as filling the special plane-cavities obtained in reverse order.

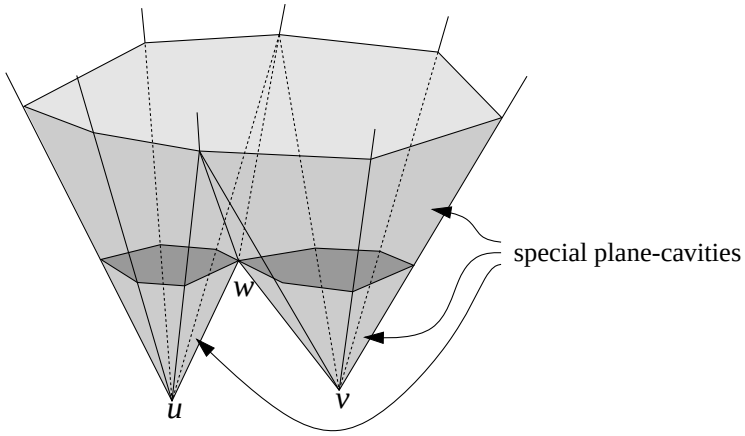


Figure 7.5: The topmost patch of lightest shade bounds a plane-cavity. This plane-cavity can be refined into a sequence of three special plane-cavities. The first two are the volume swept from u and v to w respectively. The third is the volume swept from the nearly reflex vertex w.

Theorem 15 *Let $\mathcal{P}$ be a castable polyhedron. The bounding faces of a cavity of $(\mathcal{P}, \sigma_k)$, for any sequence σ_k of special plane-cavities, receive the same color and sign.*

Proof: We prove this by induction on k. The basis step involves sweeping a plane from a reflex vertex v of $\mathcal{P}$ until a new vertex in the cavity is encountered. We show that the

faces incident to v (i.e., faces swept over) will receive the same color and sign. Let d be a feasible removal direction from which v is visible. Without loss of generality, we can assume that each face f is assigned a positive sign if $cone(f)$ contains d.

Each extreme vertex of the spherical convex hull of $\pi_{v,d}(\overline{St}(v))$ is $\pi_{v,d}(x)$ for some reflex edge vx. Consider two neighbouring extreme vertices $\pi_{v,d}(a)$ and $\pi_{v,d}(b)$. Let D_{va} and D_{vb} be the cones of directions for the color groups containing the faces incident to va and vb, respectively. By Lemma 35, $D_{va} \cap D_{vb}$ is nonempty. By Lemma 36, $D_{va} \cap \overline{D_{vb}}$ is empty. Thus, the two color groups containing the faces incident to va and vb are eligible for merging. By applying this argument to every pair of neighbouring extreme vertices of the convex hull, we conclude that the faces incident to vx for all extreme vertices $\pi_{v,d}(x)$ will receive the same color.

Next, we argue that the faces that are incident to reflex edges between neighbouring extreme vertices $\pi_{v,d}(a)$ and $\pi_{v,d}(b)$ will also receive the same color. The portion of $Lk(v)$ between va and vb projects to a bay of $\pi_{v,d}(\overline{St}(v))$. Let D be the common cone of directions associated with the faces incident to va and vb.

First, pick out all reflex edges vx between va and vb such that $cone(vx)$ does not contain $\pi_{v,d}(a)$ and $\pi_{v,d}(b)$, and neither $cone(va)$ nor $cone(vb)$ contains $\pi_{v,d}(x)$. By Lemma 37, $D \cap \overline{D_x}$ is empty where D_x is the cone of directions associated with the faces incident to vx. By Lemma 35, $D \cap D_x$ is nonempty. Thus, the coloring algorithm will eventually assign the same color for faces incident to va, vb, and all such reflex edges vx.

There are four kinds of remaining reflex edges unaccounted for. The first two kinds include reflex edges vx such that $\pi_{v,d}(x)$ lies outside $cone(va) \cup cone(vb)$, and $cone(vx)$ contains either $\pi_{v,d}(a)$ or $\pi_{v,d}(b)$. The other two kinds include reflex edges vx such that $\pi_{v,d}(x)$ lies inside $cone(va) \cup cone(vb)$.

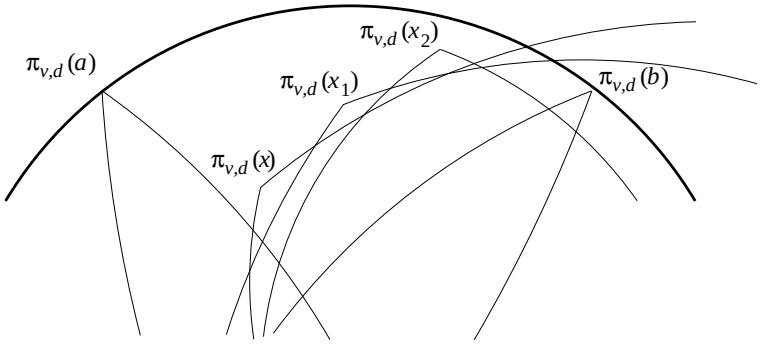


Figure 7.6: $cone(vx_k)$ does not contain $\pi_{v,d}(b)$

Suppose that $\pi_{v,d}(x)$ lies outside $cone(va) \cup cone(vb)$ and $cone(vx)$ contains $\pi_{v,d}(b)$. Then examine the reflex edge vx_1 after vx. Note that $\pi_{v,d}(x_1)$ lies outside $cone(va) \cup cone(vb)$,

and $cone(vx_1)$ does not contain $\pi_{v,d}(a)$. If $cone(vx_1)$ also contains $\pi_{v,d}(b)$, then we examine the next reflex edge vx_2 and so on. Thus, we obtain a sequence $vx, vx_1, \cdots, vx_k$ such that $cone(vx_k)$ does not contain $\pi_{v,d}(b)$ in its interior as in Figure 7.6. So the faces incident to vx_k are in the same group for *va* and *vb*. By construction, Lemma 37 is applicable to *va*, vx_{k-1}, and vx_k. Thus, together with Lemma 35, we conclude that the faces incident to vx_{k-1} will receive the same color as those incident to *va*. Now, repeat the argument for *va*, vx_{k-2}, and vx_{k-1}, and so on. Eventually, the faces incident to $vx, vx_1, \cdots, vx_{k-1}$ will receive the same color as those incident to *va* and *vb*. Similar argument works for the case where $\pi_{v,d}(x)$ lies outside $cone(va) \cup cone(vb)$ and $cone(vx)$ contains $\pi_{v,d}(a)$. This takes care of the first two kinds of remaining reflex edges.

Take a successive pair of reflex edges vy_1 and vy_2 that we have already put in the same group for *va* and *vb*. Note that $cone(vy_1)$ does not contain $\pi_{v,d}(y_2)$ and $cone(vy_2)$ does not contain $\pi_{v,d}(y_1)$. We can apply the previous reasoning to color faces incident to each reflex edge *vx* between vy_1 and vy_2 such that $\pi_{v,d}(x)$ lies outside $cone(vy_1) \cup cone(vy_2)$. By repeating this overall argument, we will assign the same color to faces incident to reflex edges between *va* and *vb* as those incident to *va* and *vb*.

Now, all the edges between a successive pair of reflex edges *vx* and *vy* are convex. Thus, if f is a face incident to such a convex edge, then $cone(f)$ contains $cone(vx) \cap cone(vy)$ and hence $cone(f)$ contains the common cone of directions, say D, for all reflex edges between *va* and *vb*. Thus, $D \cap \overline{cone}(f)$ is empty. See Figure 7.7. So if D_f is the cone of directions associated with f, $D \cap \overline{D_f}$ is also empty as $\overline{D_f} \subseteq \overline{cone}(f)$. By Lemma 35, $D \cap D_f$ is nonempty and so f will also receive the same color as those faces incident to *va* and *vb*.

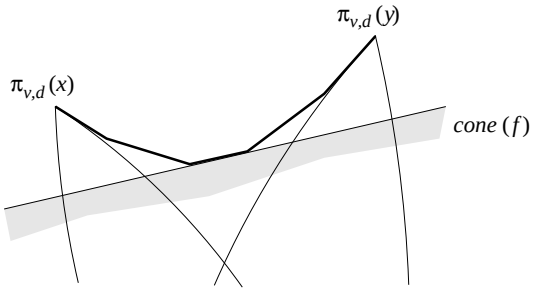


Figure 7.7: The bold polygonal chain is the projection of the link of *v* between *vx* and *vy*.

The above establishes the basis case of the induction. To proceed to the induction step, we need to associate cones of directions to the new faces introduced after filling a special plane-cavity from a reflex or nearly reflex vertex *v*. These new faces are not original polyhedron faces and we call them artificial faces. Let D be the intersection of cones of directions associated with (artificial or original) faces incident to *v*. We make D the cone of directions for all artificial faces introduced after filling this special plane-cavity. Then

in the induction step, we will sweep a plane from a nearly reflex vertex w to fill another special plane-cavity. We can use the same argument for the basis case to show that the cones of directions associated with the (artificial or original) faces incident to w satisfy the conditions for receiving the same color. Since the cones of directions for artificial faces are derived inductively from intersection of cones of directions for original faces swept in the past, we conclude that the original faces incident to w will receive the same color as other original faces swept in the past. □

7.4 An implementation for coloring faces

The first phase of the algorithm merges color groups at reflex edges. This can be easily done in $O(n)$ time by traversing the boundary of $\mathcal{P}$.

In phase 2, we merge groups G_1 and G_2 whenever $dcone(G_1) \cap dcone(G_2)$ consists of exactly two connected components. There are two cases: (1) $cone(G_1) \cap cone(G_2) \neq \emptyset$ and $cone(G_1) \cap \overline{cone}(G_2) = \emptyset$, (2) $cone(G_1) \cap cone(G_2) = \emptyset$ and $cone(G_1) \cap \overline{cone}(G_2) \neq \emptyset$. Recall that the cone of a group is the intersection of cones for each face in the group. In case (1), the condition could also be stated that there exists a direction with positive projection on all face normals in G_1 and G_2, and there is no direction with positive projection on the normals in G_1 that has negative projection on the normals in G_2. Similarly, the condition in case (2) could also be stated that there exists a direction with positive projection on the normals in G_1 that has negative projection on the normals in G_2, and there is no direction with positive projection on all face normals in G_1 and G_2.

To identify a mergeable pair, we build the arrangement of cones and their complements by building an arrangement of the n great circles that contribute to the current set of cones and their complements. Two spherical convex polygons A and B representing cones have one of four relationships: either their boundaries intersect, A is inside B, B is inside A, or they are disjoint. For a given cone A, all boundary intersections can be detected by walking the boundary of A in the arrangement. All cones including A can be found while building the arrangement by determining which cones include any chosen vertex of the boundary of A. Once all cones including cones are known, then the reverse relationship is also known, and the disjoint pairs are those that remain. If there are pairs that can be merged, at least one pair will be identified after $O(n^2)$ steps. Given a mergeable pair, it is not difficult to merge them in time $O(n^2)$. Thus, in $O(n^3)$ time we can color all faces. Since much of the above computation can be reused in subsequent steps, we suspect that this can be improved.

Theorem 16 *Given a castable polyhedron $\mathcal{P}$ of size n, the coloring algorithm assigns color and signs to faces of $\mathcal{P}$ in $O(n^3)$ time so that faces of the same color and sign belong to same mould part. Moreover, boundary faces of a cavity of $(\mathcal{P}, \sigma_k)$, for any sequence σ_k of plane-cavities, receive the same color and sign.*

Upon completion, the coloring algorithm will return several double cones of directions, and any feasible pair of opposite removal directions belongs to such a double cone. Afterwards, an eminently practical approach to identify a feasible removal direction is to select a random sample of directions from each cone and test the feasibility of these selected directions using the $O(n \log n)$-time algorithm of Chapter 3

7.5 Further applications to casting

Objects to be manufactured may also be non-castable. For example, a cube with a depression on each face is not castable using two mould parts. Such a problem is usually resolved by using *side-cores*. A side-core is an additional part. For the example of a cube with a depression on each face, we can introduce four side-cores to occupy the depression on each vertical face. The two main mould parts are in contact with the rest of the cube. During object ejection, the four side-cores are removed first, and the two main mould parts can then be removed without blockage.

There is an alternative way to define plane-cavities that facilitates the handling of side-cores. Sweep a plane from a reflex vertex of $\mathcal{P}$ until a saddle vertex (with respect to the sweeping direction) is encountered. We call the volume swept a *restricted plane-cavity* of $\mathcal{P}$. Consider the union of restricted plane-cavities of $\mathcal{P}$. Our techniques can be carried over to show that bounding faces of a connected component in the union must be removed in the same direction. Furthermore, our coloring algorithm will assign the same color and sign to such bounding faces. It is natural to assert that bounding faces of one such connected component should be occupied by a side-core or a main mould part. Thus, this helps us to identify where to use side-cores as well as their retraction directions.

Concluding remarks

We have studied the problem of determining whether there is a two-part cast for a given object such that the two cast parts can be removed without collision. We considered the case where the removal directions must be opposite, and gave necessary and sufficient conditions under which a cast exists. We also developed an algorithm to compute the cast for polyhedral objects, and the variant of this algorithm that we implemented performs fairly well in practice.

We have also studied two variants of the two-part cast problem: One of them is identical to the two-part cast problem, except that the cast machinery has a certain level of uncertainty in its directional movement. In the other one, two cast parts are to be removed in two given directions and these directions need not be opposite. For both problems, we gave complete characterizations under which a cast exists, and obtain algorithms to verify these conditions for polyhedral parts.

We defined a geometric feature, the *cavity*, which facilitates the process of analyzing manufacturability and the automated design of a cast for the object. We also provide algorithms to extract it from objects.

There are several interesting directions for further research.

While our implementation performs well on medium size models, more experimentation is necessary to develop a robust, practically useful, efficient heuristic implementation. Many objects in real life are not polyhedral, so the algorithm should be extended to handle more general object boundaries, such as cubic B-spline patches.

In Section 5.4, we provided an algorithm to construct a set of all possible pairs of direc-

tions in which the given polyhedral object is castable in time $O(n^{14}\log n)$. We consider a 4-dimensional parameter space, and construct a set of $O(n^3)$ algebraic surfaces whose arrangement has complexity $O(n^{12})$. Current algorithm takes $O(n^2\log n)$ time for each cell in the arrangement to test the castability. It would be a challenge to exploit the potential coherence between neighboring cells in order to reduce the complexity rather than paying $O(n^2\log n)$ per cell.

Another interesting issue is to study the extra possibilities that cores and inserts give.

Finally, it is desirable to maximize the "flatness" of the parting surface between the two cast parts. Majhi et al. [37] considered this problem for convex polyhedral objects. They proposed a "flatness" measure and gave an $O(n^2)$ time algorithm to find a cast that optimizes this measure, where n is the number of vertices. It would be interesting to see whether our algorithm for computing all directions of castability can be adapted so that it reports the direction allowing the flattest parting surface.

Bibliography

[1] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 517–526, 1992.

[2] H.-K. Ahn, S.-W. Cheng, and O. Cheong. Casting with skewed ejection direction. In *Proc. 9th Annu. Internat. Sympos. Algorithms Comput.*, volume 1533 of *Lecture Notes Comput. Sci.*, pages 139–148. Springer-Verlag, 1998.

[3] H.-K. Ahn, S.-W. Cheng, O. Cheong, and J. Snoeyink. Cavities and castability ananlysis. Tech. report, Institute of Information and Computing Sciences, Utrecht University, 2001.

[4] H.-K. Ahn, S.-W. Cheng, O. Cheong, and J. Snoeyink. The reflex-free hull. In *Proc. 13th Canadian Conference on Computational Geometry*, pages 9–12, 2001.

[5] H.-K. Ahn, S.-W. Cheng, O. Cheong, and J. Snoeyink. The reflex-free hull. Tech. report, Institute of Information and Computing Sciences, Utrecht University, 2001.

[6] H.-K. Ahn, O. Cheong, and R. van Oostrum. Casting with directional uncertainty. Tech. report, Institute of Information and Computing Sciences, Utrecht University, 2001.

[7] H.-K. Ahn, M. de Berg, P. Bose, S.-W. Cheng, D. Halperin, J. Matoušek, and O. Schwarzkopf. Separating an object from its cast. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 221–230, 1997.

[8] M. A. Armstrong. *Basic Topology*. McGraw-Hill, London, UK, 1979.

[9] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu. Feasibility of design in stereolithography. In *Proc. 13th Conf. Found. Softw. Tech. Theoret. Comput. Sci.*, volume 761 of *Lecture Notes Comput. Sci.*, pages 228–237. Springer-Verlag, 1993.

[10] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu. Feasibility of design in stereolithography. *Algorithmica*, 19(1–2):61–83, Sept. 1997.

[11] P. Bose. *Geometric and Computational Aspects of Manufacturing Processes*. PhD thesis, McGill University, 1994. Also available as UBC Tech Rep 95-02, Dept. of Comp. Sci, Univ. of British Columbia, 1995.

[12] P. Bose, D. Bremner, and M. van Kreveld. Determining the castability of simple polyhedra. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 123–131, 1994.

[13] P. Bose, D. Bremner, and M. van Kreveld. Determining the castability of simple polyhedra. *Algorithmica*, 19(1–2):84–113, Sept. 1997.

[14] P. Bose and G. Toussaint. Geometric and computational aspects of manufacturing processes. *Comput. & Graphics*, 18:487–497, 1994.

[15] P. Bose, M. van Kreveld, and G. Toussaint. Filling polyhedral molds. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes Comput. Sci.*, pages 210–221. Springer-Verlag, 1993.

[16] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[17] L. Chen, S. Chou, and T. Woo. Parting directions for mould and die design. *Comput. Aided Design*, 25:762–768, 1993.

[18] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. *Discrete Comput. Geom.*, 14:261–286, 1995.

[19] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[20] R. Elliott. *Cast iron technology*. Butterworths, London, UK, 1988.

[21] S. P. Fekete and J. S. B. Mitchell. Geometric aspects of injection molding. Workshop on Geometric and Computational Aspects of Injection Molding, Bellairs Research Institute, February 5–12, 1993.

[22] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4(2):74–123, Apr. 1985.

[23] S. Gupta, D. Das, W. Regli, and D. Nau. Automated manufacturability analysis: A survey. *Research in Engineering Design*, 9(3), 1997.

[24] J. Han and A. Requicha. Feature recognition from cad models. *IEEE Computer Graphics and Applications*, 18(2):80–94, 1998.

[25] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes Comput. Sci.* Springer-Verlag, June 1991.

[26] J. G. Hocking and G. S. Young. *Topology*. Addison-Wesley, Reading, MA, 1961.

[27] C. M. Hoffmann. Parametric modeling. to be published.

[28] C. M. Hoffmann. Solid modeling. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 47, pages 863–880. CRC Press LLC, Boca Raton, FL, 1997.

[29] K. Hui. Geometric aspects of mouldability of parts. *Comput. Aided Design*, 29:107–208, 1997.

[30] K. Hui and S. Tan. Mould design with sweep operations - a heuristic search approach. *Comput. Aided Design*, 24:81–91, 1992.

[31] e. James H. Marsh. *The Canadian Encyclopedia*. McClelland and Stewart, 2000. (online) http://www.thecanadianencyclopedia.com.

[32] R. Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1989.

[33] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput. Geom. Theory Appl.*, 3(3):157–184, 1993.

[34] K. K. Kwong. *Computer-aided parting line and parting surface generation in mould design*. PhD thesis, University of Hong Kong, Hong Kong, 1992.

[35] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[36] T. Lozano-Pérez, M. T. Mason, and R. Taylor. Automatic synthesis of fine-motion strategies for robots. *Internat. J. Robot. Res.*, 3(1), 1984.

[37] J. Majhi, P. Gupta, and R. Janardan. Computing a flattest, undercut-free parting line for a convex polyhedron, with application to mold design. In *Proc. 1st ACM Workshop on Appl. Comput. Geom.*, volume 1148 of *Lecture Notes Comput. Sci.*, pages 39–47. Springer-Verlag, 1996.

[38] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[39] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[40] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[41] W. I. Pribble. Molds for reaction injection, structural foam and expandable styrene molding. In J. H. DuBois and W. I. Pribble, editors, *Plastics mold engineering handbook*. Van Nostrand Reinhold Company Inc., New York, NY, 1987.

[42] D. Rappaport and A. Rosenbloom. Moldable and castable polygons. *Comput. Geom. Theory Appl.*, 4:219–233, 1994.

[43] W. Regli. *Geometric Algorithms for Recognition of Features from Solid Models*. PhD thesis, University of Maryland, 1995.

[44] W. C. Regli, S. K. Gupta, and D. S. Nau. Feature recognition for manufacturability analysis. In K. Ishii, editor, *Proc. 1994 ASME Computers in Engineering Conference*, pages 93–104. American Society of Mechanical Engineers, New York, NY, Sept. 1994.

[45] A. Rosenbloom and D. Rappaport. Moldable and castable polygons. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 322–327, 1992.

[46] J. J. Shah. Conceptual development of form features and feature models. *Research in Engineering Design*, 2:93–108, 1991.

[47] J. J. Shah and M. Mäntylä. *Parametric and Feature-Based CAD/CAM*. John Wiley and Sons, New York, 1995.

[48] M. Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete Comput. Geom.*, 12:327–345, 1994.

[49] S. S. Skiena. *The Algorithm Design Manual*. Telos/Springer-Verlag, New York, 1998.

[50] M. Soss and G. Toussaint. A hierarchy of polygons: a study of polygon properties. presentation at Canada-Cuba Workshop on Algorithms, May 2000.

[51] I. E. Sutherland. *Sketchpad: A man-machine graphical communication system*. PhD thesis, Massachusetts Inst. Tech., Cambridge, MA, Jan. 1963.

[52] S. B. Tor and A. E. Middleditch. Convex decomposition of simple polygons. *ACM Trans. Graph.*, 3:244–265, 1984.

[53] G. T. Toussaint. Movable separability of sets. In G. T. Toussaint, editor, *Computational Geometry*, pages 335–375. North-Holland, Amsterdam, Netherlands, 1985.

[54] M. van Kreveld. Person polygons: Why not? *Snapshots of Computational and Discrete Geometry*, 3, 1995. Tech. Rep. SOCS-94.50. McGill School of Comp. Sci., http://www.cs.uu.nl/people/marc/personpoly.ps.gz.

[55] C. F. Walton and T. J. Opar, editors. *Iron castings handbook*. Iron Casting Society, Inc., 1981.

Index

Acknowledgements

So many people have made a significant contribution to this thesis in various ways. Although all of them provided me with valuable information, a few played a more direct role in the preparation of this thesis.

First of all, I extend my gratitude to my promotor, Mark Overmars, and my co-promotor, Otfried Cheong for their perfect supervision. During the writing of my Ph.D. thesis, Mark Overmars provided many corrections and suggestions for improvement of the manuscript, for which I'm grateful. I am deeply indebted to Otfried Cheong: When I was a M.Sc. student at POSTECH in Korea, he motivated me to do a Ph.D. in Computational Geometry and provided me an offer of being his first Ph.D. student. With unfailing courtesy and monumental patience, he have supervised me in a perfect way, in Hong Kong and the Netherlands.

I would also like to thank Siu-Wing Cheng and Mordecai Golin. Siu-Wing Cheng had advised me in various ways, and we had worked together on several problems. Mordecai Golin had showed me interesting geometry problems that I challenged to solve.

I should also thank René van Oostrum. When we were in Hong Kong, I learned from him how to develope films and print photographs. He also helped me a lot when I settled down to the life in the Netherlands, and translated "Summary" into Dutch for this thesis.

I must acknowledge my debt to the co-authors of the papers of which this thesis is composed: apart from Otfried Cheong, Siu-Wing Cheng and René van Oostrum, these are Mark de Berg, Prosenjit Bose, Dan Halperin, Jiří Matoušek, and Jack Snoeyink.

I am no less grateful to Prosenjit Bose, Siu-Wing Cheng, Jan van Leeuwen and Peter van Emde Boas, for taking place in the reviewing committee. Prosenjit Bose and Jan van Leeuwen made detailed comments on the manuscript, which have helped me very substantially with the preparation of this thesis.

It is with particular pleasure that I express my deeply-felt gratitude to Henk and Suze

Kroon. They provided us a comfortable and great place to stay while I and my wife were staying in the Netherlands.

There is not enough space to mention all my friends who indirectly contributed to this thesis, even if I had a complete record of their contributions.

Finally, my thanks to my wife, Yu-Jin Yi, and our unborn child for their support and tolerance during the writing of this thesis.

감사의 글

1997년 9월부터 4년 남짓 수행했던 연구가 본 논문으로 결실 맺게 되어 하나님께 감사 드립니다. 본 논문이 나오기까지 여러 가지로 도움을 주신 많은 분들에게 감사의 글을 전할 수 있는 이 페이지가 논문의 가장 중요한 부분인 것 같습니다. 저의 생애에서 정말 감사한 것은, 자기 시간을 쪼개서 남을 돕고 그 자체를 좋아하는 사람들을 만난 것입니다.

홍콩에서의 유학시절 동안 저를 격려해 주시고 위로해 주신 홍콩동신교회 교우님들께, 특히 귀한 말씀으로 가르치시고 삶으로 인도하신 김성준 목사님과 백성범 목사님께 감사를 드립니다. 한가족처럼 저를 아껴주시고 사랑해 준 청년회 여러분들(변우성/박대윤/손일옥/정도원/홍순남/...), 저희 가족을 위해 멀리 네덜란드까지 귀한 한국음식을 가지고 찾아주었던 김정래/탁윤정씨, 그리고 함께 홍콩과기대학에서 생활하며 지냈던 동생들(안경진/한요섭)에게 감사를 드립니다.

네덜란드에서 지낸 1년2개월 여 동안 많은 사랑을 함께 나누었던 사랑의교회 교우님들, 처음 교회에 온 우리 부부를 따뜻하게 맞아주신 이세령 목사님, 그리고 귀한 말씀과 함께 사랑으로 섬기신 문장환 목사님과 김성진 사모님, 저희가족을 섬겨 주시고 늘 깊은 관심을 보여주셨던 이영환 장로님, 이욱현 장로님께 감사드립니다. 저희가족을 위해 수고를 아끼지 않으셨고 뛰어난 음식솜씨를 발휘해 주신 혁상씨, 늘 멋쩍은 웃음의 찬종씨와 그 가족들(소영/재욱/민수), 기쁨과 슬픔을 함께 나누었던 진희네 가족(현준/소영/진희/선희), 늘 잔잔한 미소로 저희들을 사랑해 주신 중혁/부영씨, 그리고 모든 청년부 식구들에게 감사 드립니다. Utrecht 에서 함께 유학하며 많은 도움을 주셨던 창준이네와 경우씨네 가족들에게도 감사를 드립니다.

멀리서 항상 저를 생각해 주시고 격려와 사랑을 아끼지 않으신 어머니께 누구보다 더욱 감사를 드립니다. 네덜란드에서 보낸 생활동안 이해와 사랑으로 남편을 섬겨주었고 지혜로움으로 가정을 아름답게 꾸려온 아내와, 저희 부부에게 큰 기쁨을 안겨다 준, 곧 태어날 우리 아이에게 넘치는 사랑과 감사를 전합니다. 사랑스런 아내를 낳아 길러주시고, 저를 친자식처럼 사랑해주시는 장인어른/장모님께도 감사를 드립니다.

마지막으로, 위에서 언급하진 못했지만 도움을 주신 분들께 감사를 전합니다.

Samenvatting

Bij het industrieel vervaardigen van allerlei gebruiksvoorwerpen, variërend van kookpotten tot telefoons, en van machines zoals locomotieven en vliegtuigen of onderdelen daarvan, worden verschillende technieken gebruikt. Gieten is daar één van, en deze techniek wordt met name toegepast wanneer de te maken objecten van metaal of kunststof zijn. Het gietproces bestaat uit twee stappen. Eerst wordt vloeibaar materiaal in een uit twee delen bestaande mal gegoten. Daarna, wanneer het materiaal gestold is, wordt één van de twee delen van de mal verwijderd, waarbij het object wordt meegenomen en tenslotte uit het verwijderde deel van de mal wordt gehaald. Zowel bij het verwijderen van de mal met daarin het object als bij het uitnemen van het object uit de mal moet ervoor gezorgd worden dat noch het object, noch de delen van de mal beschadigd raken, zodat de kwaliteit van het object gegarandeerd is en de mal opnieuw gebruikt kan worden. Er zijn verschillende vormen van het gietproces, onder meer afhankelijk van de gebruikte materialen (ijzer, aluminium, polymeren, zink, enzovoorts), de productie van de mal zelf (met de hand of in massaproductie), en de manier van vullen van de mal (onder invloed van zwaartekracht of onder druk).

Het onderzoeksgebied van de geometrische algoritmen is ontstaan als een onderdeel van de theoretische informatica en heeft zich ontwikkeld tot een zelfstandige discipline die zich bezighoudt met algoritmen en datastructuren voor geometrische objecten. Daarbij ligt de nadruk op algoritmen die exact zijn en een asymptotisch snelle looptijd hebben. Het gebied heeft zich in de loop der tijd in verschillende richtingen ontwikkeld, en heeft raakvlakken met gebieden als computer graphics, wetenschappelijke visualisatie, geografische informatiesystemen (GIS), geometrische modellering en computer-ondersteunde vervaardiging (CAD/CAM), robotica, virtual reality, enzovoorts. De geometrische vraagstukken uit de verschillende toepassingsgebieden vereisen zorgvuldig ontwikkelde algoritmen om tot goede en efficiënte oplossingen te komen.

Computer-ondersteund ontwerp (CAD) is een vorm van automatisering die ontwerpers ondersteunt bij het vervaardigen van tekeningen, specificaties, lijsten van onderdelen, en andere met het ontwerp-proces samenhangende taken, met behulp van grafische en rekenintensieve computerprogramma's. CAD-systemen hebben het maken van een industrieel ontwerp, de eerste fase in het ontstaan van een nieuw product, aanmerkelijk vereen-

voudigd. De producten kunnen variëren van printplaten, machine-onderdelen en meubels tot complete gebouwen. In alle gevallen is het resulterende product te beschouwen als een geometrisch object, en het is te verwachten dat zich allerlei vraagstukken van geometrische aard voordoen.

Door de geometrische aard van het industriële gietproces rijzen er veel geometrische vraagstukken bij de automatisering ervan. CAD-systemen kunnen een onwerper van een onderdeel helpen om al in de ontwerpfase te verifiëren of het onderdeel daadwerkelijk te maken is met behulp van het gietproces, zonder dat het nodig is om voor die verificatie een prototype te maken. Aan de basis van de verificatie ligt een geometrisch beslissingsprobleem: is het mogelijk om een mal te construeren voor het te maken onderdeel, zodanig dat de twee delen van de mal verwijderd kunnen worden zonder schade toe te brengen aan het onderdeel of aan elkaar? De geometrie van het onderdeel speelt, samen met de door het gietproces opgelegde beperkingen, en belangrijke rol in het beantwoorden van deze vraag. Het zou kunnen zijn dat, wanneer de mal niet zorgvuldig is ontworpen, één of beide delen van de mal niet verwijderd kunnen worden. De vraagstukken waar we naar kijken houden zich hiermee bezig: gegeven een drie-dimensionaal object, bestaat er een mal waarvan de delen verwijderd kunnen worden nadat het gegoten object is gestold. Een object waarvoor dit het geval is noemen we *castable* (gietbaar).

In dit proefschrift bestuderen we het *castability problem* (gietbaarheids-vraagstuk) in drie verschillende modellen voor gieten met gebruik van een uit twee delen bestaande mal. In het eerste model moeten de twee delen van de mal in tegenovergestelde richting worden verwijderd. We onderscheiden twee gevallen, afhankelijk van het al dan niet van tevoren gespecificeerd zijn van de richting van verwijdering. Het tweede model is vrijwel identiek aan het eerste, maar heeft als verschil dat er zekere mate van speling zit in de richting waarin de delen van de mal worden verwijderd. Ook nu maken we weer het onderscheid tussen de gevallen waarin de richting van verwijdering al dan niet van tevoren is opgegeven. In het derde model hoeven de richtingen waarin de delen van de mal worden verwijderd niet tegenovergesteld te zijn. Voor alledrie de modellen geven we voorwaarden voor castability, en ontwikkelen we algoritmen om polyhedrale objecten te testen op het voldoen aan deze voorwaarden.

Bepaalde eigenschappen van een te vervaardigen object kunnen van invloed zijn op de analyse van de castablility ervan, en op het automatisch ontwerpen van een mal voor het object. Zo beperkt een gat of deuk in het oppervlak van het object de verzameling van richtingen waarin de delen van de mal verwijderd kunnen worden. Immers, het gedeelte van de mal dat in het gat of de deuk steekt moet verwijderd kunnen worden zonder het object te beschadigen. Het herkennen van dergelijke eigenschappen kan de zoekruimte van richtingen waarin de delen van de mal verwijderd kunnen worden aanzienlijk verkleinen, en op deze manier het automatisch ontwerpen van een mal vereenvoudigen. We definiëren een geometrische eigenschap, de *cavity* (holte), die gerelateerd is aan de castability van objecten, en we geven algoritmen om cavities in objecten te herkennen.

요 약

제조(製造)란 철, 유리 혹은 폴리머와 같은 원료로 생필품(주전자, 전화기 등)에서부터 기관차, 비행기 등의 기계를 만드는 과정이다. 제조에는 사람이 노동력을 사용하여 소량의 제품을 만드는 수공업이나 기계를 이용하는 대량으로 제품을 생산하는 기계공업이 있다. 좁은 의미에서, 제조공정은 상대적으로 큰 규모의 완성품을 제작하거나 혹은 부품들을 조립하는 것이다. 주로 사용되는 제조공정 가운데 주조(鑄造)는 플라스틱이나 금속 제품을 생산하기 위해 일반적으로 사용되는 제조공정이다. 주조는 오랜 기간 동안 널리 사용되어 왔는데, 주로 가정용품, 주방용품, 기계의 몸체와 부품 등을 만드는데 이용되어 왔다. 우선 만들려고 하는 제품에 대하여 싸고도 질이 좋은 것을 주조하는 방법을 생각하여야 하는데, 이를 위해서는 주형(鑄型)을 어떻게 설계하여 조합할 것인가가 중요한 문제이다. 원칙적으로는 코어(core:中型)를 가급적 사용하지 않고 되도록 전체를 두 부분(고정주형/가동주형)으로 고안한다. 산업현장에서 이용되는 주조공정은 크게 두 단계로 구성된다. 먼저 제품의 모양을 본떠 만든 주형에 용융액이 가득 채워진다. 용융액이 단단하게 굳어 제품이 되면 주형이 열리는데, 가동주형이 제품과 함께 이동하여 고정주형으로부터 분리되고, 가동주형으로부터 제품이 반대방향으로 이동하여 주형으로부터 완전히 분리된다. 주형과 제품의 이동과정에서 제품이나 주형 모두 손상되어서는 안되며, 그리하여 생산된 제품의 질이 보장되고 주형은 제품을 생산하기 위해 다시 사용될 수 있다. 주조에는 제품생산에 사용되는 원료의 종류(철, 알루미늄, 폴리머, 아연 등), 주형의 대량생산성, 그리고 용융액 주입방식에 따라 모래 거푸집 주조, 사출성형, 다이 캐스팅 등 여러 가지 주조방식이 있다.

계산기하학(計算幾何學)은 전자계산학 이론의 한 분야로 시작하여 발전했으며 기하학적 계산을 하는 연구분야 이다. 계산기하학은 기하학적 물체에 대한 알고리즘과 자료구조의 조직적인 연구라 정의될 수 있으며 빠른 정밀 알고리즘에 구하는데 주안점을 둔다. 이 분야는 여러 방향으로 발전되어 왔는데, 특히 기하 계산과 관련한 여러 응용분야(컴퓨터 그래픽스, 컴퓨터를 이용한 시각화, 지리정보 시스템 (GIS), 기하 모델링 (CAD/CAM), 제조, 로봇공학, 가상현실 등)와 연계되어 왔다. 이러한 다양한 응용분야에서 발생하는 많은 기하학 문제는 신중하게 설계된 기하 알고리즘으로 해결될 수 있다.

캐드(CAD) 는 설계자가 설계도, 제품 명세서, 부품목록, 그리고 그 외의 설계관련 요소들을 작성하는 과정을 보조하는 자동화의 한 형태로 그래픽스와 연산을 빠르게 처리하는 특수 컴퓨터 프로그램을 사용한다. 캐드시스템은 신제품 제작의 첫 단계인 설계를 상당히 간편하게 해왔다. 캐드를 이용한 제품은 회로판, 기계부품, 가구를 비롯해 건물 등 아주 다양하다. 이 모든 경우에 생산되는 제품은 기하학적 객체로 모든 종류의 기하학 문제가 발생하게 된다.

주조공정은 근본적으로 기하학적 특성을 가지기 때문에 주조공정의 자동화에는 많은 기하학 문제가 발생하게 된다. 산업현장에서 사용되는 캐드시스템은 설계자가 제품의 설계단계에서 제품이 주조공정을 사용하여 제조 가능한지 여부를 검사할 수 있도록 할 수 있다. 시제품을 만들지 않고도 주조가능성을 확인할 수 있는 이러한 검증은 다음과 같은 기하학적 판단에 기초하고 있다 : "제품을 주형으로 둘러싼 후, 주형이 두 부분으로 나뉘어지고, 서로 충돌하지 않으면서 이동하여 제품으로부터 분리될 수 있는가?" 주조공정 자체의 몇 가지 제약과 함께 제품의 기하정보가 이 문제의 해답을 결정한다. 주형이 제대로 설계되지 않으면 주형이 이동할 때 제품에 걸려 이동하지 못하여 주조공정이 실패할 수 있다. 이 논문은 주조공정과 관련하여 바로 이 문제를 다룬다. "3차원의 물체에 대해, 충돌하지 않으면서 제품으로부터 분리될 수 있는 주형이 존재하는가?" 그러한 경우 이 물체는 주조가능(castable:鑄造可能)하다고 한다.

이 논문에서 우리는 두 부분으로 구성된 주형으로 이루어진 세 가지 다른 주조모델에 대해 각각 주조가능성 문제를 다룬다. 첫 번째 주조모델에서는 주형의 두 부분이 서로 반대 방향으로 이동하여 주형이 열린다. 이동 방향이 미리 주어지느냐의 여부에 따라 두 가지 경우를 다룬다. 두 번째 모델은 첫 번째 모델과 거의 같으며, 주형이 이동할 때 주조기계가 주어진 이동방향으로부터 어느 정도의 오차를 가지고 이동하는 경우다. 이 모델에서도 정해진 범위의 오차 내에서 이동방향이 미리 주어지느냐의 여부에 따라 두 가지 경우를 다룬다. 세 번째 모델에서는 주형의 두 부분이 각각의 이동방향으로 이동하며, 두 이동 방향은 서로 반대일 필요는 없다. 본 논문은 이 세 가지 주조 모델에서 다면체의 물체에 대한 주조가능성의 모든 특징을 제시하고, 이 특징을 검증하는 알고리즘을 설명한다.

물체의 특징점(features:特徵點)은 제조와 관련한 정보를 제시하는데, 물체의 제조가능성을 분석하는 과정과 주형의 자동설계를 용이하게 한다. 예를 들면 물체의 표면에 작은 구멍이나 움푹 들어간 곳이 있으면, 물체를 주조할 경우 구멍이나 움푹 들어간 곳을 차지한 주형이 물체와 충돌하지 않고 이동되어야 하므로, 주조가능한 주형의 이동방향의 영역를 축소시킨다. 그러한 특징점들을 인식하는 것은 설계를 용이하게 할 뿐만 아니라, 주조가능한 주형의 이동방향을 구할 때 탐색영역을 줄여준다. 또한 그런 특징점들은 설계 자동화를 원활하게 한다. 본 논문에서는 물체의 주조가능성과 관련된 기하학적 특징점 (cavity)을 정의하고, 물체로부터 이 특징점을 추출하는 알고리즘을 제시한다.

Curriculum Vitae

Hee-Kap Ahn

3 april 1973

Geboren te Dae-gu, Zuid-Korea.

maart 1992 – february 1996

Studie Computer Engineering aan de Kyungpook National University in Zuid-Korea.

maart 1996 – february 1998

Studie Computer Science aan de Pohang University of Science and Technology. (M.Sc)

september 1997 – juni 2000

Ph.D. Candidate aan de Hong Kong University of Science and Technology.

september 2000 – oktober 2001

Assistent in Opleiding aan het Informatica-instituut van de Universiteit Utrecht.