Separating an Object from its Cast

Hee-Kap Ahn* Mark de Berg† Prosenjit Bose‡ Siu-Wing Cheng*
Dan Halperin§ Jiří Matoušek¶ Otfried Schwarzkopf*

Abstract

In casting, liquid is poured into a cast that has a cavity with the shape of the object to be manufactured. The liquid then hardens, after which the cast is removed. We consider the case where the cast consists of two parts and address the following problems: (1) Given a cast for an object and a direction $\vec{d}$, can the cast be partitioned into two parts such that the parts can be removed in directions $\vec{d}$ and $-\vec{d}$, respectively, without colliding with the object or the other cast part? (2) How can one find a direction $\vec{d}$ such that the above cast partitioning can be done? We give necessary and sufficient conditions for both problems, as well as algorithms to decide them for polyhedral objects. We also give some evidence that the case where the cast parts need not be removed in opposite directions is considerably harder.

keywords: Casting, molding, cast removal, cast design, separability, polyhedral terrains.

1 Introduction

Background. The *design-for-manufacturing* and *design-for-assembly* paradigms suggest that objects should be designed in such a way that manufacturing and assembly can be performed easily, and therefore cheaply. To support this, computer-aided design systems have to be augmented with a component verifying *on-line* that an object being designed can actually be manufactured using the intended techniques. Algorithms in such a verification component are required to work purely on the basis of a CAD model of the object. Such algorithms have been proposed for a number of manufacturing processes, such as injection molding [5, 10, 24], NC-machining [13], and stereolithography [1]. For an overview of the geometric problems arising in these manufacturing processes, the reader is referred to Bose's thesis [2] and the survey by Bose and Toussaint [4].

In assembly, the emphasis is on planning tasks so as to put parts together to form the final product. Interesting geometric problems arise in almost every step of automatic assembly planning. Assembly sequencing [27], part orienting [11], fixturing [21], and welding [19], are just a few of many examples.

Although these manufacturing and assembly processes may be different from one another, some of them raise similar geometric problems that can be generally termed *separability* problems [25]. This is the case in the manufacturing process that we study in this paper: the process of casting [9, 26].

The casting process consists of filling the cavity between two cast parts with some molten liquid. After the liquid has solidified, the cast parts are removed and one is left with an object whose shape is that of the cavity. The directions in which the cast parts are removed are called the *removal directions* or *parting directions*. Figure 1 illustrates the process on a 2-dimensional example. The key property necessary for

*Department of Computer Science, HKUST, Clear Water Bay, Hong Kong. Supported partly by the Research Grant Council CERG HKUST 6074/97E.

†Department of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands. Supported by the Netherlands' Organization for Scientific Research (N.W.O.) and by ESPRIT Basic Research Action No. 7141 (project ALCOM II: *Algorithms and Complexity*).

‡School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada, K1S 5B6. Research supported in part by NSERC. E-mail: jit@scs.carleton.ca

§Department of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. Research supported by an Alon Fellowship, by ESPRIT IV LTR Project No. 21957 (CGAL), and by the Hermann Minkowski – Minerva Center for Geometry at Tel Aviv University.

¶Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic. Supported by Czech Republic Grant GAČR 0194 and by Charles University Grants No. 193,194.

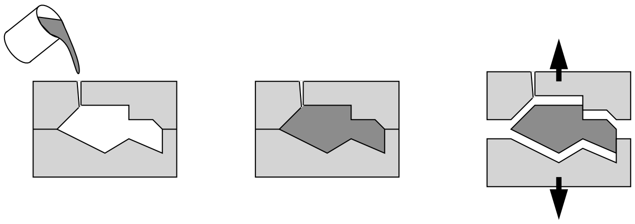

Figure 1: The casting process.

casting is that the cast parts can be removed from the object without destroying either the cast parts or the object. This ensures that the given object can be mass produced by re-using the same cast parts. In the example of Figure 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.

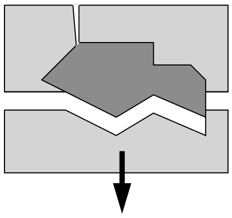

Figure 2: The top part of the cast is stuck.

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 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 [9, 23, 26]—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, in the sense that if they were not removed, then the cast could not be removed without breaking. However, their use slows down the manufacturing process and makes it more costly. We do not study the extra possibilities of cores and inserts.

Related work. The 2-dimensional version of our problem has been studied by Rappaport and Rosenbloom [24]. 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 [15] gave a heuristic approach to the 3-dimensional problem. Some candidate 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 [16] gave an algorithm to determine the feasibility of a given parting direction. 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. [7] 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 though one exists.

Hui [14] 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. [3] 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 [3]. 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 [18] 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 results. Our paper is mainly 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 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 4 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 4). 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 5) 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 4 we give an exact algorithm, which computes all combinatorially distinct directions 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.

Finally, in Section 6 we briefly study the case where the cast parts need not be removed in opposite directions. Here we present the following surprising result, which shows that this case is a lot more difficult than the opposite-removal case. In sand casting, the problem of finding a good cast reduces to the problem of finding a plane subdividing the boundary of the object into two terrains. Extending the curve splitting the object boundary to a surface splitting the cast in two removable parts is trivial: one simply splits the cast with the plane defining the curve. In our more general setting, the object boundary must also be split into two terrains, but now the curve doing so need not lie in a plane. Hence, it is no longer straightforward to extend the curve to a surface that splits the cast into two removable parts. In fact, we show that that this extension is not always possible for the case of arbitrary removal directions: there are polyhedra whose boundary can be decomposed into two terrains but that are not castable. (For opposite cast removal we prove that it is always possible to extend a curve splitting the object boundary into two terrains to a surface splitting the cast in removable parts.) This implies that Hui and Tan's [15] and Kwong's [16] approaches are not sufficient for finding a legal cast with arbitrary removal directions. It has to be noted, though, that we are not aware of any industrial setting in which non-opposite directions are being used for cast removal, and we therefore do not study this case in more detail.

We also give a necessary and sufficient condition for an object to be castable in two given (not necessarily opposite) directions. Unfortunately, the condition does not always lead to a two-part cast: the portion of the cast that is removed in the second removal direction may consist of several pieces.

[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.

2 Preliminaries: Assumptions and Terminology

Throughout this paper, $\mathcal{P}$ 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—they should be merged into one—but we do allow coplanar non-adjacent facets. 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. For a more thorough description of polyhedra and some of their properties, the reader is referred to the book by Preparata and Shamos [22].

The characterization of castability that we give in the next section applies to both polyhedra and curved objects. This is important since many industrial parts are not polyhedral. More precisely, we shall be dealing with objects $\mathcal{B}$ 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 $\mathcal{B}$ be topologically equivalent to a polyhedron $\mathcal{P}$ as defined above: it must be a solid whose boundary is a connected 2-manifold.

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 cavity whose shape is that of the object to be manufactured is contained in the interior of B. As stated in the introduction, we are interested in casts consisting of two parts. One of them will be called the *red part* and is denoted by $\mathcal{C}_r$, the other will be called the *blue part* and is denoted by $\mathcal{C}_b$.

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

The *combinatorial complexity* of the arrangement $\mathcal{A}(\mathcal{Q})$ is the total number of faces (of all dimensions) in the arrangement. If Q consists of q curves, each being a great circle or an arc of a great circle, then the complexity of the arrangement is $O(q^2)$ and it can be $\Omega(q^2)$. We say that the curves in $\mathcal{Q}$ are in *general position* if no three curves in $\mathcal{Q}$ 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).

We call an object $\mathcal{B}$ *$\vec{d}$-monotone* if every line with direction $\vec{d}$ intersects the interior of $\mathcal{B}$ 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. Hence, it is z-monotone.

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

3 Opposite Cast Removal—Testing a Direction

In this section we present a criterion for testing whether a given object $\mathcal{B}$ 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 $\mathcal{B}$ 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 $\mathcal{B}$ 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 $\mathcal{B}$ 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 $\mathcal{B}$ 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 3.1 *An object $\mathcal{B}$ is castable if and only if it is vertically monotone.*

Proof: Assume that $\mathcal{B}$ 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 $\mathcal{B}$, and let p and q be two points in $\ell \cap \mathrm{int}(\mathcal{B})$. Since a point $r \in \ell$ in between p and q can be translated neither upward nor downward without colliding with $\mathcal{B}$, the point r can be in neither $\mathcal{C}_r$ nor $\mathcal{C}_b$. Hence $r \in \mathcal{B}$. We must even have $r \in \mathrm{int}(\mathcal{B})$; otherwise there would be a point $r' \notin \mathcal{B}$ having a point $p' \in \mathcal{B}$ above it and a point $q' \in \mathcal{B}$ below it—this follows from p and q being in the interior of $\mathcal{B}$—and such a point r' can be in neither $\mathcal{C}_r$ nor $\mathcal{C}_b$. This proves that $\mathcal{B}$ is vertically monotone.

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

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 illustrates that vertical facets sometimes need to be colored partly red and partly blue. In the figure, the

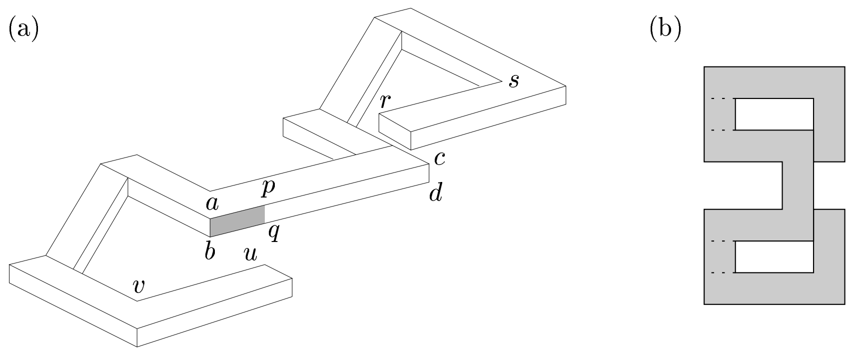

Figure 3: A polyhedron with a facet that needs two different colors. (a) A 3D view of the polyhedron. (b) The projection onto the xy-plane of the polyhedron, with the interior shaded.

facet $abcd$ is coplanar with the vertical facets incident to 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 3.1 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 3.1 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, 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 3.2 *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 4 shows the polygon we get when we project the polyhedron of Figure 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 [8], for

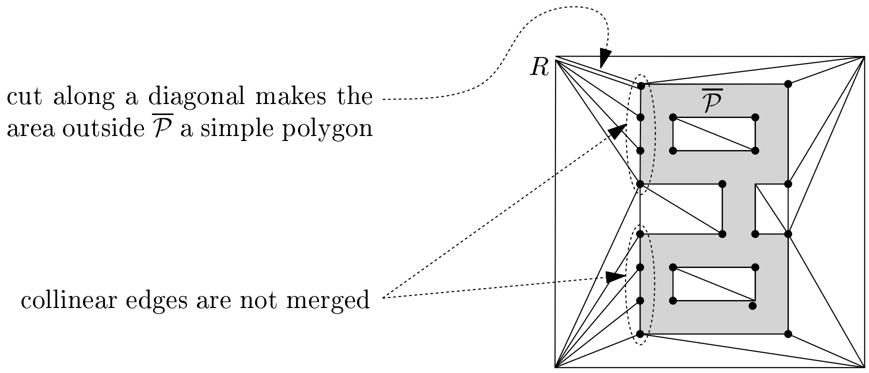

Figure 4: The projection of the polyhedron of Figure 3, and a triangulation of its complement.

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 [22] 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 projection of a vertex of $\mathcal{P}$. With each vertex $\overline{v}$ of $\overline{\mathcal{P}}$, we can therefore associate a vertex v of $\mathcal{P}$ that projects onto $\overline{v}$. 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 [6]. 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 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 $\overline{v}$ 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 $\overline{t} \in \overline{\sigma}$ that shares an edge $\overline{e}$ 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 $\overline{t}$ 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 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 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 5. Furthermore, if a triangle $\overline{t}$ shares an edge $\overline{e}$ 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 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. $\square$

To check a polyhedron $\mathcal{P}$ for castability, we can use Lemma 3.1 and the following simple observation.

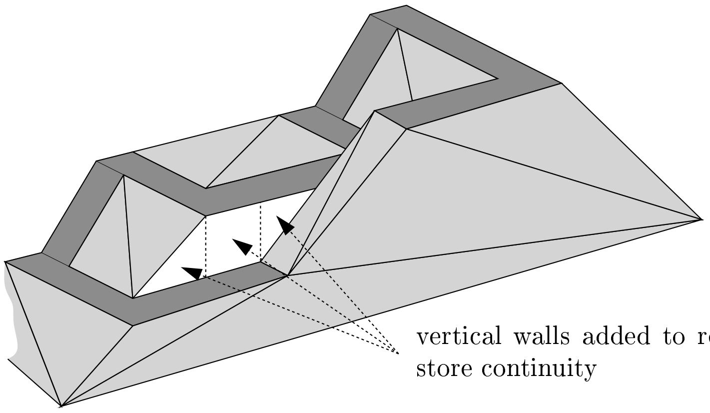

Figure 5: A discontinuity in the surface σ^*.

Observation 3.3 *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 [16]—see the Introduction and immediately implies Theorem 3.9. 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, silhouette edges increase 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 $\mathcal{B}$ be an object, and consider a vertical line ℓ. The line ℓ intersects $\partial\mathcal{B}$ in a number of maximal closed intervals. These intervals separate open intervals lying either completely inside or outside $\mathcal{B}$. A boundary interval that is surrounded on both sides by intervals outside $\mathcal{B}$ is called a *convex silhouette interval*. A boundary interval that is surrounded on both sides by intervals in the interior of $\mathcal{B}$ 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 $\mathcal{B}$.

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 consists of *all* vertical facets of the polyhedron. Figure 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.

The following lemma is straightforward:

Lemma 3.4 *If the reflex silhouette of an object $\mathcal{B}$ is not empty, then $\mathcal{B}$ 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 $\partial\mathcal{B}$, 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.

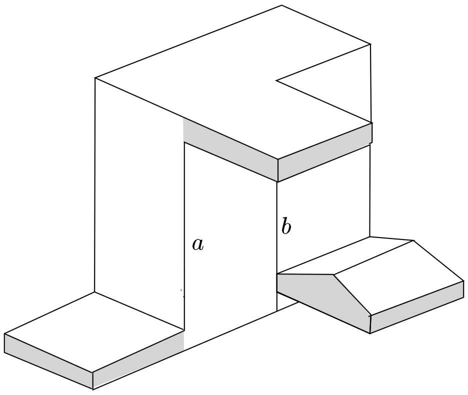

Figure 6: The silhouette of a polyhedron with vertical facets.

Lemma 3.5 *Let $\mathcal{B}$ be an object with empty reflex silhouette and such that no vertical line contains two silhouette intervals. Then $\mathcal{B}$ is vertically monotone.*

Proof: Let S be the silhouette of $\mathcal{B}$. Since no vertical line contains two silhouette intervals, the shadow curves of $\mathcal{B}$ 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, $\ldots$, R_{k+1}, every one of which is topologically equivalent to a disc with a finite number of holes, as in Figure 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.

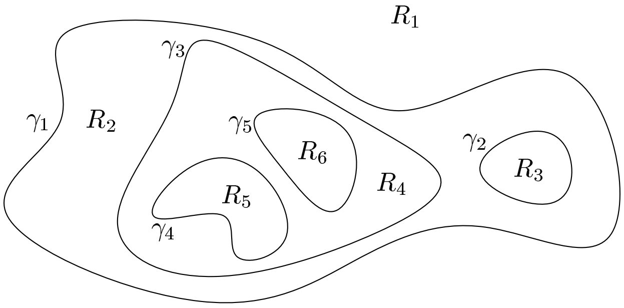

Figure 7: A collection of shadow curves, and the regions they define.

For a point p in the xy-plane, let $\ell(p)$ denote the vertical line through p. The set $\ell(p) \cap \mathcal{B}$ is a disjoint union of closed intervals. We number those intervals from bottom to top as $I_1(p)$, $I_2(p)$, $\ldots$, $I_m(p)$. 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 $p \in \mathcal{B} \setminus C$ to a point in C. However, this violates the fact that the $\mathcal{B}$ is connected. Therefore, there can only be one curve, implying that $\mathcal{B}$ is vertically monotone.

$\sqcup\!\sqcap$

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, 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(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 $\mathcal{B}$ 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 8: Shrinking the shadow curve of Figure 3(b).

We can now prove the generalization of the previous lemma.

Lemma 3.6 *Let $\mathcal{B}$ be an object with empty reflex silhouette and non-crossing shadow curves. Then $\mathcal{B}$ is vertically monotone.*

Proof: Assume that $\mathcal{B}$ is not vertically monotone. Then there is a vertical line ℓ containing two points p and q in the interior of $\mathcal{B}$, and a point r between p and q outside or on the boundary of $\mathcal{B}$. Let $\varepsilon > 0$ be such that the balls of radius 2ε centered at p and q are contained in $\mathcal{B}$. We shrink $\mathcal{B}$ to obtain a new object $\mathcal{B}'$ by removing from $\mathcal{B}$ every point that has distance at most ε to the complement of $\mathcal{B}$. If ε is chosen small enough, this results in a legal object $\mathcal{B}'$, that is, a solid whose boundary is a connected 2-manifold. Since p and q lie in the interior of $\mathcal{B}'$ and r lies outside $\mathcal{B}'$ and between p and q, the object $\mathcal{B}'$ is not vertically monotone. However, if the silhouette of $\mathcal{B}$ consists of non-crossing convex silhoutte curves, then the silhouette of $\mathcal{B}'$ consists of disjoint convex silhoutte curves because of the shrinking. But then Lemma 3.5 implies that $\mathcal{B}'$ 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 3.7 *Let $\mathcal{B}$ be an object with empty reflex silhouette whose shadow curves are crossing. Then $\mathcal{B}$ 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 $\mathcal{B}$ 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 $\mathcal{B}$. A slight perturbation of this line shows that there exists a point outside $\mathcal{B}$ lying between two points in the interior of $\mathcal{B}$. Therefore, $\mathcal{B}$ is not vertically monotone.

□

We summarize Lemmas 3.4, 3.6, and 3.7 in the following theorem.

Theorem 3.8 *An object $\mathcal{B}$ 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 3.8, 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 9. The four cases are as follows:

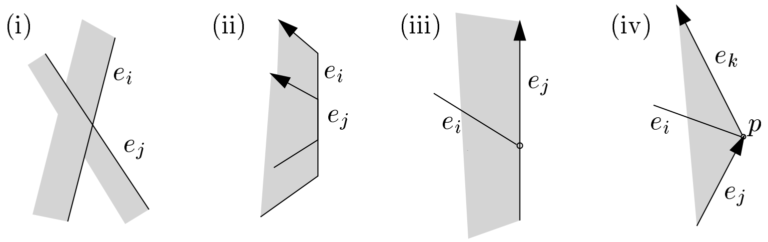

Figure 9: Four different ways in which shadow curves can cross.

(i) the interiors of e_i and e_j intersect;

(ii) e_i and e_j overlap and have 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.

Note that the condition on the orientation in (ii) ensures that the shadow curve in Figure 8 is correctly labeled as non-crossing.

By combining Theorem 3.8 with the characterization of crossing edges, we can compute efficiently whether a polyhedron is castable.

Theorem 3.9 *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. $\sqcup\!\!\!\sqcap$

4 Opposite Cast Removal—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 3.1 and Observation 3.3 $\mathcal{P}$ 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 circle. 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), 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 3.3 and Theorem 3.9. 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) 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., [20].) 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 [12]).

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 by a plane-sweep algorithm. By Theorem 3.8, 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 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 sweep line efficiently in this case is described in detail in [8, Chapter 2]. It remains to handle the counters update 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 sweep over all 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 4.1 *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 4.2 *There exist polyhedra for which there are $\Omega(n^4)$ distinct directions in which there is a good cast.*

Proof: Figure 10 shows a polyhedron with two horizontal "legs" and there is 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 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 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 $\Omega(n^4)$. $\square$

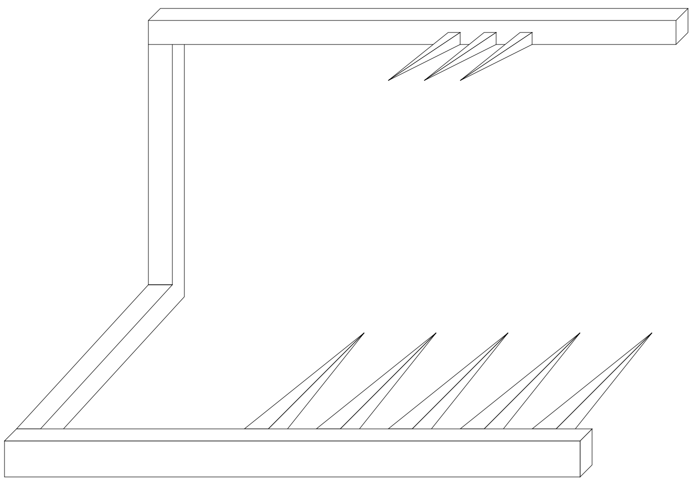

Figure 10: A component of the polyhedron having $\Omega(n^4)$ distinct cast directions

5 Experimental Results

We have implemented a simplified version of the algorithm of Theorem 3.9 (instead of a plane sweep, we simply test all pairs of shadow edges for crossings). Given an object 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 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.

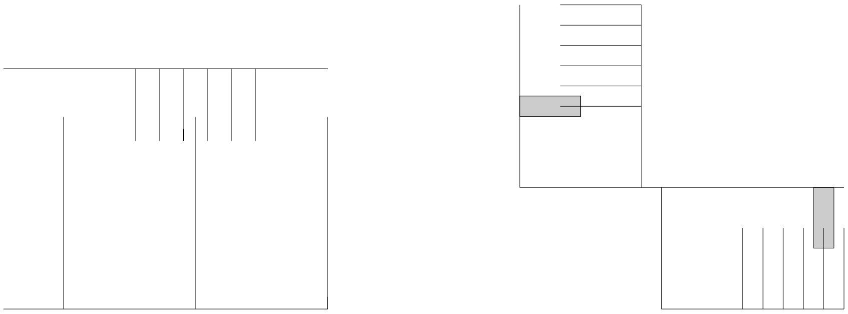

Figure 11: A top view of the lower bound construction

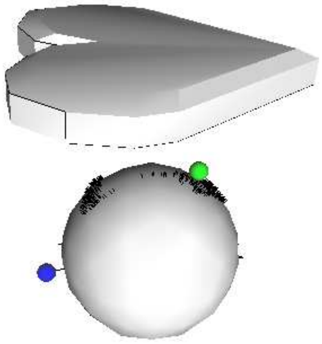

Figure 12: A heart-shaped object and the sphere of directions depicting the different casting directions tested

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 the two marked directions (the randomly chosen one).

The rook from a chess game in Figure 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.

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

6 Arbitrary Cast Removal

We now briefly discuss the case where the removal directions for the cast parts are not necessarily opposite. We call a polyhedron $\mathcal{P}$ castable with cast $\mathcal{C} = (\mathcal{C}_r, \mathcal{C}_b)$ and removal directions $\vec{d_r}, \vec{d_b}$ if $\mathcal{C}_r$ can be translated to infinity in direction $\vec{d_r}$ without collisions with $\mathcal{C}_b$ or $\mathcal{P}$, and $\mathcal{C}_b$ can be translated to infinity in direction $\vec{d_b}$ without collisions with $\mathcal{P}$. Contrary to the case of opposite removal directions, the order of removing the casts can be important. The cast induces a partition of $\partial\mathcal{P}$ into two parts, a red part touching $\mathcal{C}_r$ and a blue part touching $\mathcal{C}_b$. When looking at $\mathcal{P}$ from infinity in direction $-\vec{d_r}$ every point on the red part must be visible. Similarly, every point on the blue part must be visible when looking from infinity in direction $-\vec{d_b}$. In other words, the red and the blue part of $\partial\mathcal{P}$ must each form a terrain. The reverse of this statement, however, is not true, as the next theorem shows.

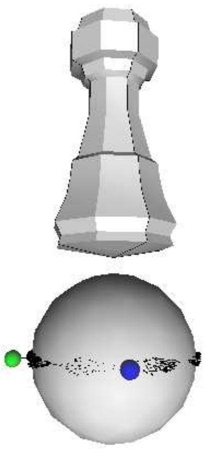

Figure 13: A rook and the sphere of directions depicting the different casting directions tested

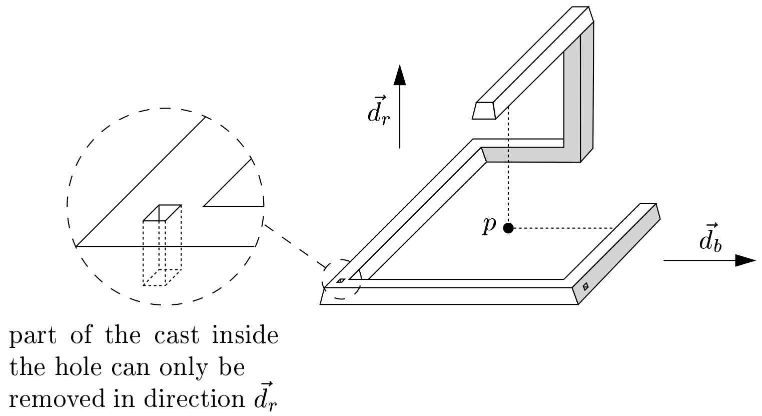

Figure 14: Illustration for the proof of Theorem 6.1.

Theorem 6.1 *In the case of arbitrary cast removal, there are polyhedra that are not castable even though their boundary can be split into two terrains with respect to the removal directions.*

Proof: Consider the polyhedron $\mathcal{P}$ and the removal directions depicted in Figure 14. The shaded facets of $\mathcal{P}$ together with the bottom facets form the blue part of the boundary; the remaining facets form the red part. Both the blue and the red part are terrains in their respective removal directions, as can be seen in Figure 15. 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 $\vec{d}_b, \vec{d}_r$. By poking two thin holes into the object, as in Figure 14, we can ensure there cannot be a good cast for any other pair of directions either. $\square$

7 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

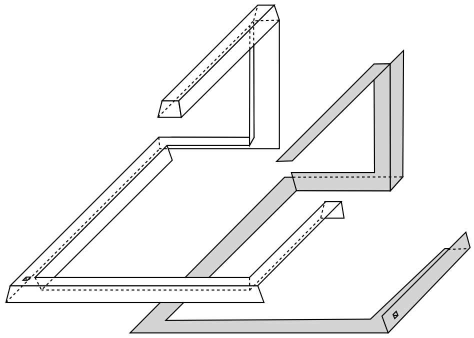

Figure 15: The boundary partition of the polyhedron of Figure 14.

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 briefly touched on the case of removal directions that are not necessarily opposite, and showed that in that case a polyhedron may not be castable even if its boundary can be split into two terrains with respect to the removal directions.

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.

Furthermore, it would be useful 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. [17] 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.

References

[1] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu. Feasibility of design in stereolithography. *Algorithmica: Special Issue on Manufacturing*, 19:61–83, 1997.

[2] 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.

[3] P. Bose, D. Bremner, and M. van Kreveld. Determining the castability of simple polyhedra. *Algorithmica: Special Issue on Manufacturing*, 19:84–113, 1997.

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

[5] Prosenjit Bose, Marc van Kreveld, and Godfried Toussaint. Filling polyhedral molds. *Computer-Aided Design: Special Issue on Manufacturing*, 30(4): 245–254, 1998.

[6] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6:485–524, 1991.

[7] L.L. Chen, S.Y. Chou, and T.C. Woo. Parting directions for mould and die design. *Computer-Aided Design*, 25:762–768, 1993.

[8] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

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

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

[11] Kenneth Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[12] Leonidas J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4:74–123, 1985.

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

[14] K. Hui. Geometric aspects of mouldability of parts. *Computer Aided Design*, 29(3):197–208, 1997.

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

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

[17] J. Majhi, P. Gupta, and R. Janardan. Computing a flattest, undercut-free parting line for a convex polyhedron, with application to mold design. In *ACM Workshop on Applied Computational Geometry*, pages 39–45, 1996.

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

[19] M. McAllister and J. Snoeyink. Two-dimensional computation of the three-dimensional reachable region for a welding head. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 437–442, 1993.

[20] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[21] Mark Overmars, Anil Rao, Otfried Schwarzkopf, and Chantal Wentink. Immobilizing polygons against a wall. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 29–38, 1995.

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

[23] 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.

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

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

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

[27] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71:371–396, 1994.