Casting with Skewed Ejection Direction*

Hee-Kap Ahn[1] Siu-Wing Cheng[2] Otfried Cheong[3]

Abstract

Casting is a manufacturing process in which liquid is poured into a cast (mould) that has a cavity with the shape of the object to be manufactured. The liquid then hardens, after which the cast is removed. We address geometric problems concerning the removal of the cast. A cast consists of two parts, one of which retracts in a given direction carrying the object with it. Afterwards, the object will be ejected from the retracted cast part. In this paper, we give necessary and sufficient conditions to test the feasibility of the cast part retraction and object ejection, where retraction and ejection directions need not be the same. For polyhedral objects, we show that the test can be performed in $O(n^2 \log^2 n)$ time and the cast parts can be constructed within the same time bound. The complexity of the cast parts constructed is worst-case optimal. We also give a polynomial time algorithm for finding a feasible pair of retraction and ejection directions for a given polyhedral object.

1 Introduction

The manufacturing industry has at its disposal a number of processes for constructing objects, including gravity casting, injection molding [6, 16], stereolithography [3], NC-machining [10]. In all of these manufacturing contexts, computer-aided design systems of growing sophistication are presently being introduced, and more and more real-world objects are modeled as geometric objects within a computer. These systems have to be augmented with a component verifying, purely on the basis of a CAD model of the object, that an object being designed can actually be manufactured using the intended techniques. The survey by Bose and Toussaint [4, 7] gives an overview of geometric problems and algorithms arising in these manufacturing processes.

The casting process [9, 17] consists of two stages. First, liquid is poured into a cavity formed by two cast parts. After the liquid hardens, the cast is opened and the manufactured object is removed from the cavity. In plastic injection molding machinery, this second stage is often implemented as follows: one of the two cast parts is retracted mechanically, carrying the object with it. The object is then ejected from the retracted cast part, typically by a burst of compressed air.

Previous work on the casting problem has assumed that the object is ejected in the direction opposite to the retraction direction of the movable cast part. This is, in fact, not required by the existing technology for injection molding: the air burst will eject the object as long as a direction *exists* in which it can move out of the cast. Exploiting this possibility allows to cast more parts, or to cast parts with simpler moulds, and is the subject of the present paper.

To simplify our discussion, we will pretend that it is not the manufactured object that is ejected from the moving cast part, but that the cast part is removed from the object. In this way,

*This research was supported by the Research Grant Council, Hong Kong, China (project no. HKUST6074/97E) and Brain Korea 21 program of MOE. Part of it was done when H.-K. A. and O. C. were at HKUST.

[1] Department of EECS, Korea Advanced Institute of Science & Technology, Email: heekap@hanmail.net.
[2] Department of Computer Science, Hong Kong University of Science & Technology, Email: scheng@cs.ust.hk.
[3] Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands. Email: ocheong@win.tue.nl.

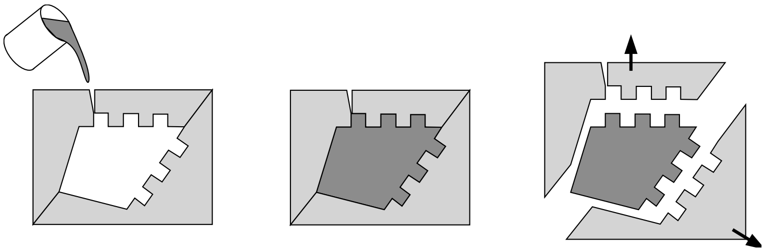


Figure 1: The casting process

both retraction and ejection are modelled conceptually by the removal of a cast part. To model the retraction, the *fixed* cast part will first be removed in a direction opposite to the retraction. To model the ejection, the remaining cast part will then be removed in a direction opposite to the ejection. Figure 1 illustrates the process on a 2-dimensional example.

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. Note that 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 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*.

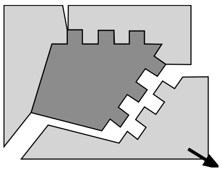


Figure 2: The top part of the cast is stuck.

Separating a cast in two arbitrary removal directions in 2D [16] and in some special 3D cases has been studied before [5]. 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 [9, 15, 17]. 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.

The 2-dimensional version of the castability problem has been studied by Rappaport and Rosenbloom [16]. 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 [12] gave a heuristic approach to the 3-dimensional problem, assuming opposite directions for cast removal. It is based on testing candidate directions using sample points, and may not find a feasible direction, or may incorrectly return an infeasible direction. Kwong [13] gave the first complete 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. Chen et al. [8] showed how to find the parting direction that maximizes the number of completely visible “cavities” in the object. This direction, however, may not be a good parting direction even if one exists. Based on Chen et al.'s work, Hui [11] gave exponential time algorithms that also take cores and inserts into account. Again they are not guaranteed to find a feasible cast. Bose et al. [5] 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 this model [5].

Finally, Ahn et al. [2] gave, to our knowledge, the first complete algorithm to determine the castability of polyhedral parts for opposite directional cast removal. Given a simple polyhedron with n vertices, they presented an $O(n \log n)$-time algorithm to compute castability in a given direction. They also presented an $O(n^4)$-time algorithm to compute all *combinatorially distinct directions* for which there is a good cast. They also showed that there exist polyhedra for which there are $\Omega(n^4)$ combinatorially distinct directions in which there is a good cast.

In this paper we give a characterization of castability, under the assumption that the cast has to consist of two parts that are to be removed in two given, not necessarily opposite directions. This characterization applies to rather general, not necessarily polyhedral objects. This is important since many industrial parts are not polyhedral. The characterization is constructive and translates directly into an algorithm. In a CAD/CAM-system that supports boolean set operations and sweeping, for instance by a volume representation, the characterization can be implemented easily.

In terms of asymptotic complexity, it seems hard to implement it efficiently, though. We therefore give an algorithm to verify castability of polyhedral objects that runs in time $O(n^2 \log^2 n)$, and produces cast parts of complexity $O(n^2)$. We show that this is optimal by giving an example of an object that cannot be cast using cast parts of lower complexity.

Results for opposite cast parts removal [2, 12, 13] rely on the property that an object is castable if and only if its boundary is visible completely from the two opposite removal directions. This is not true when the removal directions are non-opposite: there are polyhedra whose boundary is wholly visible from the removal directions but which are not castable with respect to those directions [2].

For completeness, we also give an $O(n^{14} \log^2 n)$-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.

2 A characterization of castability

We assume that the outer shape of the cast equals a box denoted by $\mathcal{B}$, which is large enough so that the object $\mathcal{Q}$ to be manufactured is contained strictly in its interior. Our goal is to decompose the cast into two parts which only overlap along their boundaries. The cast part to be removed first is called the *red part* and is denoted by $\mathcal{C}_r$. The other cast part is called the *blue part* and is denoted by $\mathcal{C}_b$. The removal directions for $\mathcal{C}_r$ and $\mathcal{C}_b$ are $\vec{d}_r$ and $\vec{d}_b$, respectively. We call $\vec{d}_r$ the *red direction* and $\vec{d}_b$ the *blue direction*. It will be convenient to define the object $\mathcal{Q}$ as a topological open set, and the cast parts $\mathcal{C}_r$ and $\mathcal{C}_b$ as closed sets. The union of $\mathcal{C}_r$ and $\mathcal{C}_b$ equals $\mathcal{B} \setminus \mathcal{Q}$. We use $\mathcal{I}$ to denote the interface $\mathcal{C}_r \cap \mathcal{C}_b$.

We denote the interior of a set $A \subseteq \mathbb{R}^3$ by $\mathrm{int}(A)$ and its closure by $\mathrm{cl}(A)$. A may be open, closed, or neither. To represent A, we store its boundary $\mathrm{bd}(A)$ which is defined as $\mathrm{cl}(A) \cap \mathrm{cl}(\mathbb{R}^3 \setminus A)$. When $\mathrm{cl}(A)$ is polyhedral, $\mathrm{bd}(A)$ consists of vertices, edges, and facets. We call them the vertices, edges, and facets of A. They may not consist of points in A in general, for example, if A is an open set.

We call an object $\mathcal{Q}$ *castable* with respect to $(\vec{d}_r, \vec{d}_b)$ if we can translate $\mathcal{C}_r$ to infinity in direction $\vec{d}_r$ without collision with $\mathcal{Q}$ and $\mathcal{C}_b \setminus \mathcal{I}$, and can then translate $\mathcal{C}_b$ to infinity in direction $\vec{d}_b$ without collision with $\mathcal{Q}$. The order of removal is important.

We require $\mathcal{C}_b$ to be a connected subset of $\mathcal{B}$. This connectivity requirement automatically holds for $\mathcal{C}_r$, as any connected component of $\mathcal{C}_r$ is guaranteed to be connected with a facet of $\mathcal{B}$.

As mentioned in the introduction, previous work on the casting problem used a visibility metaphor, which will also be helpful in our problem. Consider $\mathcal{Q}$ illuminated by two sources of parallel light. The red light source is at infinity in direction $\vec{d}_r$; 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 ray starting at p with direction $\vec{d}_r$ does not intersect $\mathcal{Q}$, and similarly define points illuminated by blue light. Note that since we assume $\mathcal{Q}$ to be open, a light ray will not stop when it grazes the boundary of $\mathcal{Q}$.

We define the *red shadow volume* $\mathcal{V}_r$ to be the set of points of $\mathcal{B} \setminus \mathcal{Q}$ not illuminated by red light, and similarly define the *blue shadow volume* $\mathcal{V}_b$.

If we sweep $\mathcal{V}_b$ to infinity in direction $\vec{d}_r$, we will encounter a set of points in $\mathcal{B}$ that we denote by $\mathcal{V}_b^*$. Note that $\mathcal{V}_b^*$ includes $\mathcal{V}_b$ itself.

Lemma 1 *If $\mathcal{Q}$ is castable, then $\mathcal{V}_r \subseteq \mathcal{C}_b \setminus \mathcal{I}$ and $\mathcal{V}_b^* \subseteq \mathcal{C}_r \setminus \mathcal{I}$.*

Proof. By definition, $\mathcal{B} \setminus \mathcal{Q} = \mathcal{C}_r \cup \mathcal{C}_b$ 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 $\mathcal{Q}$. So p cannot be a point in the red cast part $\mathcal{C}_r$. Thus, $\mathcal{V}_r \subseteq \mathcal{C}_b \setminus \mathcal{I}$. By similar analysis, $\mathcal{V}_b \subseteq \mathcal{C}_r \setminus \mathcal{I}$. Since $\mathcal{Q}$ is castable, $\mathcal{C}_r$ can be translated first to infinity in direction $\vec{d}_r$ without colliding with $\mathcal{Q}$ or $\mathcal{C}_b \setminus \mathcal{I}$. Since $\mathcal{V}_b \subseteq \mathcal{C}_r \setminus \mathcal{I}$, we conclude that $\mathcal{V}_b^* \subseteq \mathcal{C}_r \setminus \mathcal{I}$. ◻

We are now ready to prove our characterization of castability.

Theorem 2 *Given an object $\mathcal{Q}$ and removal directions ($\vec{d}_r$,$\vec{d}_b$), $\mathcal{Q}$ is castable if and only if $\mathcal{V}_r$ lies in one connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$.*

Proof. First, we prove that the condition is necessary. Since $\mathcal{Q}$ is castable, $\mathcal{V}_b^* \subseteq \mathcal{C}_r \setminus \mathcal{I}$ and $\mathcal{V}_r \subseteq \mathcal{C}_b \setminus \mathcal{I}$ by Lemma 1. Since $\mathcal{C}_b \subseteq \mathcal{B} \setminus ((\mathcal{C}_r \setminus \mathcal{I}) \cup \mathcal{Q})$, we have $\mathcal{V}_r \subseteq \mathcal{C}_b \subseteq \mathcal{B} \setminus ((\mathcal{C}_r \setminus \mathcal{I}) \cup \mathcal{Q}) \subseteq \mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$. Therefore, if $\mathcal{V}_r$ does not lie in one connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$, then neither does $\mathcal{C}_b$. This implies that $\mathcal{C}_b$ is not connected, a contradiction.

Second, we prove the sufficiency of the condition. We let $\mathcal{C}_b$ be the connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$ that contains $\mathcal{V}_r$. Then let $\mathcal{C}_r$ be $\mathcal{B} \setminus (\mathcal{Q} \cup (\mathcal{C}_b \setminus \mathcal{I}))$. Clearly $\mathcal{C}_b$ 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 $\mathcal{Q}$. This translation of $\mathcal{C}_r$ cannot be obstructed by $\mathcal{C}_b \setminus \mathcal{I}$. Otherwise, a point p in int($\mathcal{C}_r$) can see a point q in $\mathcal{C}_b \setminus \mathcal{I}$ in direction $\vec{d}_r$. If the line segment pq contains a point in $\mathcal{V}_b^*$, then q also belongs to $\mathcal{V}_b^*$, which is a subset of $\mathcal{C}_r$ by construction. Thus, $q \in \mathcal{C}_r$, which contradicts the fact that $q \in \mathcal{C}_b \setminus \mathcal{I}$. If the line segment pq does not contain any point in $\mathcal{V}_b^*$, then pq lies in $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$. Since $q \in \mathcal{C}_b$, p also belongs to the connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$ containing $\mathcal{V}_r$. Thus, $p \in \mathcal{C}_b$, which contradicts the fact that $p \in \text{int}(\mathcal{C}_r)$.

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

The condition in Theorem 2 implies that $\mathcal{V}_r \cap \mathcal{V}_b$ is empty, but is slightly stronger. Figure 3 shows an object that is not castable, even though $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. It is impossible to find a connected blue cast part for this object.

3 Testing castability

The characterization of castability of Theorem 2 is constructive and translates directly to an algorithm. In a CAD/CAM-system that supports boolean set operations and sweeping, for instance by a volume representation, the characterization can be implemented easily.

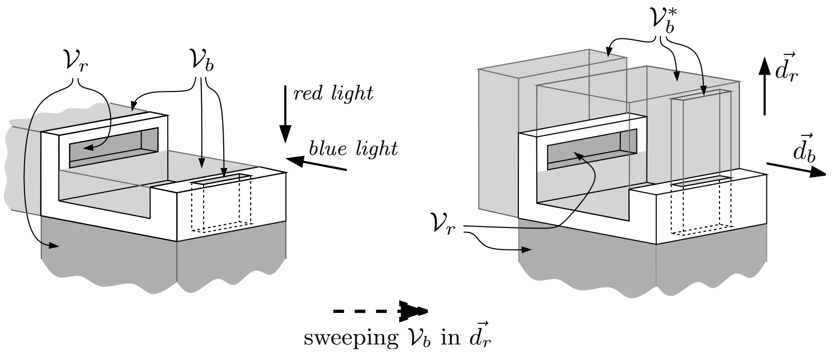
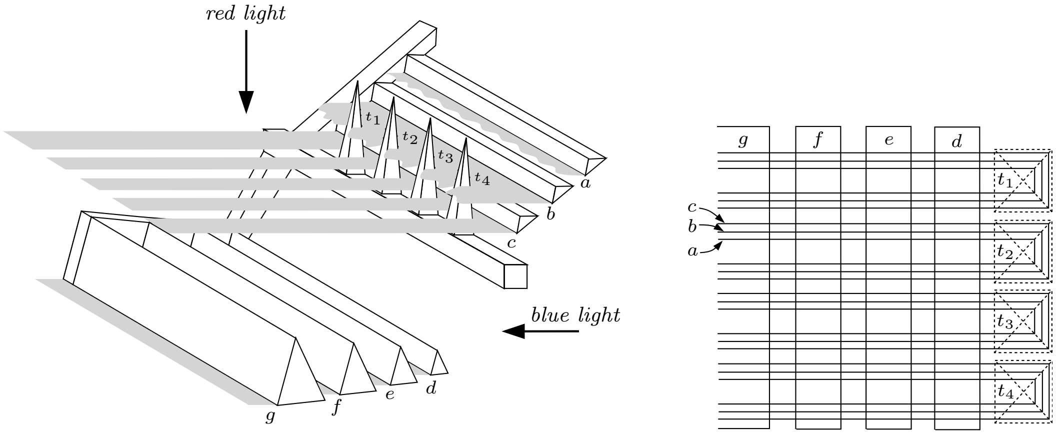


Figure 3: An object $\mathcal{Q}$ and its shadow volume. $\mathcal{V}_r$ intersects two connected components of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{Q})$.

From a more theoretical point of view, the situation is more complicated. The shadow volumes $\mathcal{V}_b$ and $\mathcal{V}_r$ can be computed by computing the visibility map of $\mathcal{Q}$. This can be done in time $O(n^2 \log n)$, if $\mathcal{Q}$ is a polyhedron with n vertices. The sweep volume $\mathcal{V}_b^*$ can then be obtained by computing the lower envelope of the boundary facets of $\mathcal{V}_b$ with respect to direction $\vec{d}_r$. In general, the lower envelope of n^2 triangles can have complexity $\Theta(n^4)$. Figure 4(a) shows some lower facets of $\mathcal{V}_b$. The bar with pyramids standing on it is particularly interesting. The lower facet of $\mathcal{V}_b$ generated by the lower facet of the bar is intentionally removed from the figure. This causes the envelope to have complexity $\Theta(n^3)$ as shown in Figure 4(b). If this removed facet is included, the complexity of the envelope lowers to $\Theta(n^2)$. This implies that computing the lower envelope by standard techniques such as divide-and-conquer or (randomized) incremental construction would need at least cubic time.



(a) (b)

Figure 4: $\Theta(n^3)$ complexity.

In the following, we give an algorithm that tests castability of an n-vertex polyhedron for a given pair of removal directions $(\vec{d}_r, \vec{d}_b)$ in time $O(n^2 \log^2 n)$. This relies on a bound of $O(n^2)$ on the complexity of $\mathcal{V}_b^*$, which we can prove under the assumption that $\mathcal{V}_b \cap \mathcal{V}_r \cap \mathrm{bd}(\mathcal{Q})$ is empty. Our algorithm therefore starts by testing this assumption.

Throughout the rest of this paper, the object to be manufactured will be an open set $\mathcal{P}$ where $\mathrm{cl}(\mathcal{P})$ is a polyhedron, that is, a possibly nonconvex solid bounded by a piecewise linear surface. The vertices, edges, and facets on this surface form $\mathrm{bd}(\mathcal{P})$. We require $\mathrm{bd}(\mathcal{P})$ to be a connected 2-manifold. Each facet of $\mathrm{bd}(\mathcal{P})$ 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—this is no restriction, as they can be merged into one—but we do allow coplanar non-adjacent facets. We also assume that $\mathcal{P}$ is *simple*, which means that no two non-adjacent facets share a point. Our assumptions imply that $\mathcal{P}$ may contain tunnels, but no voids—a polyhedron with a void is not castable anyway.

Throughout this section, we treat $\vec{d}_r$ as the positive vertical direction. We define the *red shadow* $\mathcal{S}_r := \mathcal{V}_r \cap \mathrm{bd}(\mathcal{P})$, and the *blue shadow* $\mathcal{S}_b := \mathcal{V}_b \cap \mathrm{bd}(\mathcal{P})$.

For each polyhedron edge e, let $h_b(e)$ denote the plane through e and parallel to $\vec{d}_b$. Then 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 $\mathcal{P}$ 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 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 $\mathcal{P}$, or hit an edge or facet parallel to $\vec{d}_b$ and below $\mathcal{P}$ locally, or reach infinity in direction $-\vec{d}_b$. The union of these intercepted or unintercepted blue rays define 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 $\mathcal{P}$. Upper blue curtains are symmetrically defined for upper blue silhouette edges.

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 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 $\mathrm{bd}(\mathcal{P})$

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 that $\mathcal{S}_r \cap \mathcal{S}_b = \emptyset$. Once this test is passed, we then check whether $\mathcal{V}_r \cap \mathcal{V}_b = \emptyset$. If this test is passed, we construct the cast parts and verify that $\mathcal{C}_b$ is connected.

3.1 Testing emptiness of $\mathcal{S}_r \cap \mathcal{S}_b$

The emptiness of $\mathcal{S}_r \cap \mathcal{S}_b$ can be tested in $O(n^2 \log n)$ time as follows. Let $\vec{v}$ be a vector orthogonal to both $\vec{d}_r$ and $\vec{d}_b$. Let H be a plane above $\mathrm{cl}(\mathcal{P})$ spanned by $\vec{v}$ and $\vec{d}_b$. We compute the projection of $\mathrm{cl}(\mathcal{P})$ 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 $\mathrm{bd}(\mathcal{P})$ 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 $\mathrm{bd}(\mathcal{P})$ illuminated by red light can be done in time $O(n^2 \log n)$. Similarly, we can determine the parts of $\mathrm{bd}(\mathcal{P})$ illuminated by blue light in time $O(n^2 \log n)$. We can then decide whether $\mathcal{S}_r \cap \mathcal{S}_b$ is empty by testing the intersection separately on every facet in $\mathrm{bd}(\mathcal{P})$, for instance with a plane sweep algorithm. In total, this test takes time $O(n^2 \log n)$.

3.2 Lower envelope of blue shadow facets and lower blue curtains

Once we know that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty, we examine the lower envelope, denoted by $\mathcal{L}$, of blue shadow facets and lower blue curtains. The reason is that we can compute $\mathcal{V}_b^*$ from $\mathcal{L}$. Here and in the following, let $\ell(p)$ denote the vertical line through a point p.

Lemma 3 *Let $\mathcal{L}^*$ be the set of points in $\mathcal{B}$ encountered while we sweep $\mathcal{L}$ to infinity vertically upward. Then $\mathcal{L}^* = \mathrm{cl}(\mathcal{V}_b^*)$.*

Proof. Let p be a point in $\mathcal{L}^*$, and let q be the point $\ell(p) \cap \mathcal{L}$. By definition, q is 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 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 $\mathcal{L}$ and q is in $\mathcal{L}^*$. $\square$

We show below that the complexity of $\mathcal{L}$ is $O(n^2)$. We first need two technical lemmas.

Lemma 4 *Suppose that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. For any blue shadow facets f_1 and f_2 there is no vertical line ℓ that stabs both $\mathrm{int}(f_1)$ and $\mathrm{int}(f_2)$.*

Proof. Assume to the contrary that the vertical line ℓ stabs both $\mathrm{int}(f_1)$ and $\mathrm{int}(f_2)$. Without loss of generality, assume that $\ell \cap f_1$ is below $\ell \cap f_2$. Since $f_2 \subseteq \mathrm{bd}(\mathcal{P})$, all points of ℓ lying below $\ell \cap f_2$ can not get any red light. Thus, $\ell \cap f_1 \in \mathcal{S}_r \cap \mathcal{S}_b$, a contradiction. $\square$

Lemma 5 *Suppose that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. Then a finger edge of a lower blue curtain is divided by upper blue silhouette edge(s) into segments of two types. One type consists of boundary edges of some blue shadow facets. Segments of the other type lie strictly above $\mathcal{L}$.*

Proof. Consider a finger edge of a lower blue curtain $\Gamma(e)$. It can be divided into $O(n)$ segments of two types: segments that lie on facets of $\mathcal{P}$ parallel to $\vec{d}_b$, and segments that do not. Figure 5 shows segments of each type. The segments of the former type are shadow facet edges. It remains

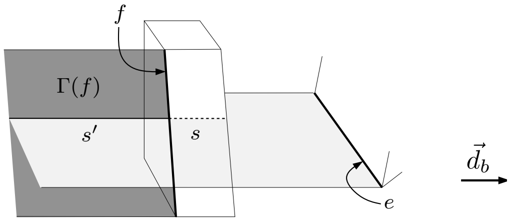


Figure 5: A finger edge consisting of two segments: s lying on a facet parallel to $\vec{d}_b$, and s' lying on $\Gamma(f)$.

to consider the segments of the latter type. It is the intersection of $\Gamma(e)$ and another blue curtain, say $\Gamma(f)$. If $\Gamma(f)$ is a lower blue curtain, then one of the two facets σ incident to f is a red shadow facet. Since a lower blue curtain $\Gamma(e)$ intersect σ, σ contains a point in $\mathcal{S}_r \cap \mathcal{S}_b$, a contradiction. So $\Gamma(f)$ must be an upper blue curtain. Since points in the interior of upper blue curtains do not appear in $\mathcal{L}$, they lie strictly above $\mathcal{L}$. $\square$

Lemma 6 *Suppose that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. The lower envelope $\mathcal{L}$ formed by all lower blue curtains and all blue shadow facets has complexity $O(n^2)$.*

Proof. The facets of $\mathcal{L}$ are bounded by three different kinds of edges: shadow facet edges including heads and tails of lower blue curtains, side edges, and finger edges. The complexity of $\mathcal{L}$ is determined by the number of vertices formed by them.

By Lemma 4, two shadow facet edges cannot define a new vertex of $\mathcal{L}$. By Lemma 5, the finger edges do not introduce any new vertex in $\mathcal{L}$. It remains to bound the number of vertices 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 $\mathrm{bd}(\mathcal{P})$ in $-\vec{d}_b$ direction, h intersects a linear number of shadow facets in a linear number of non-intersecting line segments. Consider the 2-dimensional lower envelope of the s_i's and e on h. This envelope has linear complexity. Since there are $O(n)$ side edges, $\mathcal{L}$ has $O(n^2)$ vertices in total. ◻

3.3 Computing $\mathcal{V}_b^*$

We present efficient algorithms to test the castability and construct the cast parts. We need a technical lemma.

Lemma 7 *Suppose that $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. Then for any two blue silhouette edges e and f and a vertical line ℓ intersecting e and $\mathrm{int}(\Gamma(f))$, $\ell \cap e$ is not above $\ell \cap \Gamma(f)$.*

Proof. Assume to the contrary that there is a vertical line ℓ such that $\ell \cap e$ is above $\ell \cap \Gamma(f)$. We can shift ℓ slightly such that it stabs the interior of $\Gamma(f)$ and a facet incident to e. Thus, ℓ intersects $\mathcal{P}$ above $\ell \cap \Gamma(f)$. If $\Gamma(f)$ is an upper blue curtain, then this implies that an arbitrarily 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 an arbitrarily short segment above $\ell \cap \Gamma(f)$ does not receive red or blue light. In either case, $\mathcal{V}_r \cap \mathcal{V}_b \neq \emptyset$, a contradiction. ◻

Let $\vec{v}$ be a vector orthogonal to both $\vec{d}_r$ and $\vec{d}_b$. Let H be a plane spanned by $\vec{v}$ and $\vec{d}_b$, and let $\pi(x)$ be the vertical projection of a point x onto H. The following lemma gives a necessary and sufficient condition for $\mathcal{V}_r \cap \mathcal{V}_b = \emptyset$, under the assumption $\mathcal{S}_r \cap \mathcal{S}_b = \emptyset$.

Lemma 8 *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, $\ell(p) \cap e$ is not below $\ell(p) \cap \Gamma(f)$ for some point p in $\pi(e) \cap \mathrm{int}(\pi(\Gamma(f)))$ or $\pi(e) \cap \pi(\xi(f))$.*

Proof. We prove sufficiency first. Assume to the contrary that $\mathcal{V}_r \cap \mathcal{V}_b$ is empty. Suppose that p is a point in the intersection of $\pi(e)$ and $\mathrm{int}(\pi(\Gamma(f)))$ such that $\ell(p) \cap e$ is not below $\ell(p) \cap \Gamma(f)$. By Lemma 7, $\ell(p) \cap e$ is also not above $\ell(p) \cap \Gamma(f)$. Thus, $\ell(p) \cap e = \ell(p) \cap \Gamma(f)$. The interior of $\Gamma(f)$ would contain the point $\ell(p) \cap e$. By definition, no boundary point of $\mathcal{P}$ below $\mathcal{P}$ locally can lie in the interior of a lower blue curtain, a contradiction.

The remaining alternative is that $\pi(e)$ touches $\pi(\xi(f))$ at a point p where $\ell(p) \cap e$ 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 $\mathrm{int}(\Gamma(f))$ since $\mathcal{P}$ is above σ locally. This implies that there is a short segment on ℓ above $\mathrm{int}(\Gamma(f))$ 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 $\mathcal{V}_r \cap \mathcal{V}_b$ is parallel to $\vec{d}_r$ or $\vec{d}_b$. At least one facet σ of $\mathcal{V}_r \cap \mathcal{V}_b$ is parallel to $\vec{d}_b$ and bounds $\mathcal{V}_r \cap \mathcal{V}_b$ from below, otherwise

$\mathcal{V}_r \cap \mathcal{V}_b$ would be unbounded. The facet σ cannot receive any red light as it bounds the black shadow volume from below. Thus, σ cannot be a blue shadow facet, 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 $\text{bd}(\mathcal{P})$ at a point $\nu(z)$. Suppose that we move z in the direction $-\vec{d}_b$. The vertical distance of $\nu(z)$ from $\Gamma(f)$ is monotonically decreasing and remains non-negative. Otherwise, there would be a position such that $\nu(z)$ becomes a point in the black shadow which is impossible. (See Figure 6 (a)) Before or just when $\ell(z)$ stabs an edge g of $\xi(f)$, $\nu(z)$ 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 6 (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 6 (c)) Clearly, $\pi(e)$ either intersects the interior of $\pi(\Gamma(f))$ or touches $\pi(\xi(f))$ at the intersection of $\ell(z)$ and the xy-plane. ◻

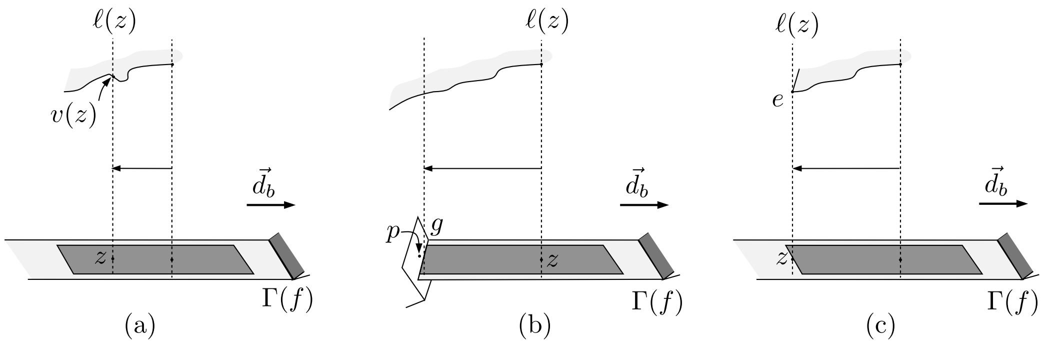


Figure 6: (a) $\nu(z)$ in the black shadow, (b) A point p in the black shadow, and (c) A lower silhouette edge e, where $\ell(z) \cap e$ does not lie below $\ell(z) \cap \Gamma(f)$

We now show how to test the emptiness of $\mathcal{V}_r \cap \mathcal{V}_b$ and constructs $\mathcal{L}$ in $O(n^2 \log^2 n)$ time. The computation aborts once it detects that $\mathcal{V}_r \cap \mathcal{V}_b \neq \emptyset$—the polyhedron is not castable in that case anyway. Once $\mathcal{L}$ is available, we can construct $\mathcal{L}^* = \text{cl}(\mathcal{V}_b^*)$. Afterwards, we can exclude the facets that lie on lower blue curtains to obtain $\mathcal{V}_b^*$. All can be done in $O(n^2 \log^2 n)$ time.

Lemma 9 *Suppose that $\mathcal{S}_r \cap \mathcal{S}_b$ is empty. In time $O(n^2 \log^2 n)$ and space $O(n^2 \log n)$ we can decide whether $\mathcal{V}_r \cap \mathcal{V}_b$ is empty, and, if so, compute $\mathcal{L}$.*

Proof. Recall that we assume that $\vec{d}_r$ is the positive vertical direction. Let $\vec{v}$ be a vector orthogonal to both $\vec{d}_r$ and $\vec{d}_b$. Let H be a plane below $\mathcal{P}$ spanned by $\vec{v}$ and $\vec{d}_b$. We first project all blue shadow facets $\mathcal{S}_b$ downward onto H. By Lemma 4, the projections of blue shadow facets are interior-wise disjoint. It follows that the complexity of the projection is $O(n^2)$. All blue shadow facets can be identified in $O(n^2 \log n)$ time by computing the complement of the part of $\text{bd}(\mathcal{P})$ which is illuminated by blue light. Edges of blue shadow facets are edges, parts of edges of $\mathcal{P}$, parts of finger edges of blue curtains, or edges in the tails of blue curtains.

We partition H into slabs by drawing lines parallel to $\vec{d}_b$ through all projected vertices of $\mathcal{P}$. These lines intersect with the projections of blue shadow facets. 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, there are $O(n^2)$ intersections which can be computed in $O(n^2 \log n)$ time. Thus, after introducing the slabs, the partitioned projection of blue shadow facets still has complexity $O(n^2)$.

We identify the upper and lower blue silhouette edges in $O(n)$ time. For each lower blue silhouette edge, we construct its lower blue curtain by intersecting a plane with $\mathcal{P}$ in $O(n \log n)$ time. So we can construct all lower blue curtains in $O(n^2 \log n)$ time.

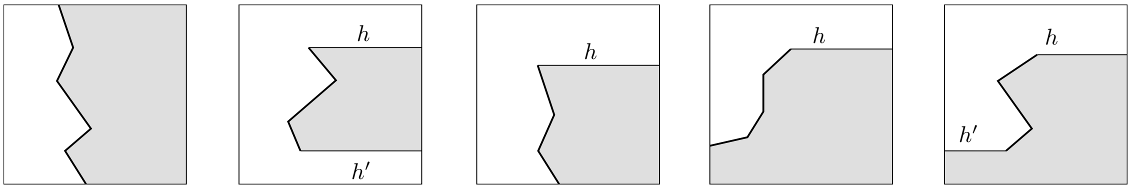


Figure 7: The boxes represent Δ_i and $\vec{d}_b$ is the rightward direction. The shaded areas represent the projected lower blue curtains. The bold chains are the p-tails. h and h' are the p-fingeredges. In the first four figures, the p-tails are also the extended p-tails. In the last figure, h' and the p-tail form the extended p-tail.

Consider a slab. We cut it into regions along the projections of upper and lower blue silhouette edges. Since upper and lower blue silhouette edges are boundary edges of blue shadow facets, we do not introduce any new vertex. The regions in the slab are linearly ordered in $-\vec{d}_b$. We label the region unbounded to the infinity in $\vec{d}_b$ by Δ_0, the next by Δ_1, and so on. We denote the boundary between Δ_i and Δ_{i-1} by ζ_i. A region Δ_i can be completely empty, or completely or partially covered by the projections of blue shadow facets. If $\mathcal{V}_r \cap \mathcal{V}_b$ is empty, no blue shadow facet is above any lower blue curtain. Thus, if we can identify the lowest blue curtain covering each empty area in every Δ_i, we obtain $\mathcal{L}$. Our algorithm carries out this computation inductively by processing $\Delta_1, \Delta_2, \ldots$ in this order. We need to maintain a segment tree $\mathcal{T}$ for storing intervals as we scan the regions.

Assume that we have come to Δ_i. Let strip(ζ_i) denote the vertical strip through ζ_i. Inductively, $\mathcal{T}$ stores the intervals at the intersections of strip(ζ_i) and the lower blue curtains that were encountered before and cover some point in ζ_i. So the intervals in $\mathcal{T}$ are in 3D. Each interval I is stored at the nodes in $\mathcal{T}$ representing the projection of I on ζ_i. The intervals stored at a node of $\mathcal{T}$ are ordered in the vertical direction. Each lower blue curtain has $O(n)$ finger edges. So there are $O(n^2)$ intervals in $\mathcal{T}$ at any time. We explicitly maintain the interval in $\mathcal{T}$, low$(\mathcal{T})$, that has the lowest endpoint among all intervals in $\mathcal{T}$. If ζ_i is the projection(s) of some lower blue silhouette edge(s), we use Lemma 8 to test the emptiness of $\mathcal{V}_r \cap \mathcal{V}_b$. If low$(\mathcal{T})$ is lower than the highest edge that projects onto ζ_i, by Lemma 8, $\mathcal{V}_r \cap \mathcal{V}_b \neq \emptyset$. So $\mathcal{P}$ is non-castable and we abort. Afterwards, we proceed to fill the empty areas in Δ_i. If ζ_i is the projection(s) of some tail edge(s), we delete from $\mathcal{T}$ the intervals induced by the corresponding lower blue curtains. Then we insert into $\mathcal{T}$ the intervals induced by the lower blue curtains of the lower blue silhouette edges that project onto ζ_i. Each insertion into $\mathcal{T}$ takes $O(\log^2 n)$ time.

For each empty area inside Δ_i, we pick one of its boundary vertices as its representative vertex. We are to fill an empty area with the lowest blue curtain covering its representative vertex. This is like a batched point location problem. We call the projection of each tail clipped within Δ_i a *p-tail*, and the projection of each finger edge clipped within Δ_i a *p-fingeredge*. We first discuss the structure of p-tails and p-fingeredges. Observe that no two p-tails cross each other by Lemma 4. If a p-tail has an endpoint in the interior of Δ_i, some p-fingeredge h is attached to this endpoint. If h points towards $-\vec{d}_b$, as the projections of upper blue silhouette edges are also used in cutting slabs into regions, Lemma 5 implies that h is the boundary edge of a projected blue shadow facet, and h connects the p-tail and ζ_{i+1}. In this case, h cannot cross any other p-tail (as a p-tail bounds projected blue shadow facets). If h points towards $\vec{d}_b$, then h connects the p-tail and ζ_i. In this case, h may cross some other p-tail. To each p-tail, we add the p-fingeredge pointing towards $-\vec{d}_b$, if there is any, to form the *extended p-tail*. No two extended p-tails cross each other. Figure 7 shows some examples of extended p-tails.

We construct a planar subdivision $\mathcal{K}_i$ using the extended p-tails as follows. From each endpoint of an extended p-tail, we extend two segments in $\vec{d}_b$ and $-\vec{d}_b$ until they hit the nearest extended p-tails or the boundary of Δ_i. $\mathcal{K}_i$ is like a trapezoidal map of the extended p-tails, and it can be

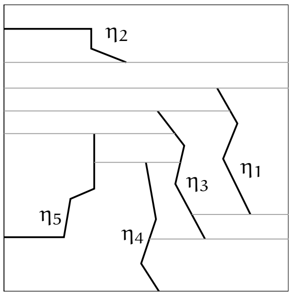


Figure 8: The box is Δ_i and $\vec{d}_b$ is the rightward direction.

built using a plane-sweep algorithm. Figure 8 shows an example. Given two extended p-tails η and η', η' is *behind* η if (1) there is a directed line pointing towards $\vec{d}_b$ that hits η before η', or (2) there is another extended p-tail η'' such that η' is behind η'' and η'' is behind η. The behind relation is a partial order. We use $\mathcal{K}_i$ to totally order the extended p-tails in a way consistent with the behind relation. We first construct a directed acyclic graph G. In $\mathcal{K}_i$, if an extension from η towards $\vec{d}_b$ hits η' or an extension from η' towards $-\vec{d}_b$ hits η, we add a directed edge from η' to η. Then we topologically sort G into a total order $\eta_1, \eta_2, \ldots$, so that if η_r is behind η_s, then $r < s$. We add ζ_i as η_0 and ζ_{i+1} as η_∞. For each face σ of $\mathcal{K}_i$, take η_r with the least index that bounds σ, and we call r the *key* of σ. We put the faces in $\mathcal{K}_i$ into a face-queue in non-decreasing order of their keys. Let m be the number of vertices inside Δ_i. The above construction runs in $O(m \log m)$ time.

Now, we are ready to traverse the faces of $\mathcal{K}_i$ to fill the empty areas in Δ_i. For each $r \geq 1$, η_r has a corresponding interval I_r stored in the segment tree $\mathcal{T}$ (except possibly for η_∞). Throughout the traversal, we maintain the smallest index r^* of the intervals in $\mathcal{T}$ induced by extended p-tails. The traversal consists of a main loop, which iterates until the face-queue becomes empty. We remove the face σ from the face-queue with the minimum key r. If $r = r^*$, we delete I_{r^*} from $\mathcal{T}$ and set $r^* = r + 1$. Afterwards, we fill the empty areas that have representative vertices inside σ as follows. Let ν be such a representative vertex. Project ν in $\vec{d}_b$ to a point ν' on strip(ζ_i). We query $\mathcal{T}$ to find the lowest interval I above ν'. Then we fill the empty area represented by ν with the lower blue curtain corresponding to I. In each iteration, the manipulation of the face-queue and $\mathcal{T}$ takes $O(1)$ and $O(\log^2 n)$ time, respectively. So the traversal of $\mathcal{K}_i$ takes $O(m \log^2 n)$ time.

Why is this filling procedure correct? Let A be the empty area represented by ν. First of all, the face-queue guarantees that by the time we consider A, the intervals in $\mathcal{T}$ for extended p-tails behind σ have been deleted. Therefore, a lower blue curtain covers ν if and only if it has an interval in $\mathcal{T}$ above ν'. Let Γ be the curtain returned by $\mathcal{T}$ that is used to fill A. It suffices to show that: (1) the p-fingeredges for Γ do not cross A, and (2) Γ is the lowest blue curtain above any point in A. Assume to the contrary that (1) or (2) is false. Then there is a lower blue curtain Γ' below Γ that covers some point in A: if (1) is false, Γ' exists by Lemma 5; if (2) is false, Γ' exists by assumption. If Γ' covers A entirely, then the querying of $\mathcal{T}$ should not have returned Γ as Γ' is a better answer, a contradiction. So some p-fingeredge for Γ' crosses A. Lemma 5 implies that there is a lower blue curtain Γ'' below Γ' that covers some point in A. But then we can replace Γ' by Γ'' and repeat the above argument. Since there is a finite number of lower blue curtains, we will reach a contradiction eventually.

After filling the empty areas, we update $\mathcal{T}$ to prepare for processing Δ_{i+1}. All the current intervals in $\mathcal{T}$ can be retained. We reexamine all extended p-tails η_r whose intervals have been deleted from $\mathcal{T}$ when traversing $\mathcal{K}_i$. If η_r intersects ζ_{i+1}, we insert into $\mathcal{T}$ the interval at the intersection of strip(ζ_{i+1}) and the lower blue curtain corresponding to η_r. This takes $O(m \log^2 n)$ time.

Since we spend $O(m\log^2 n)$ time in Δ_i, the total time needed to fill all empty areas in all slabs is $O(n^2\log^2 n)$ as the total number of vertices in all slabs is $O(n^2)$. The sum of sizes of the planar subdivisions constructed for all regions is $O(n^2)$. When processing a slab, since the number of intervals stored in $\mathcal{T}$ is $O(n^2)$, $\mathcal{T}$ uses $O(n^2\log n)$ space. ◻

3.4 Cast part construction

The computations so far have established that $\mathcal{V}_r \cap \mathcal{V}_b$ is empty and have constructed a representation of $\mathcal{V}_b^*$. This implies that $\mathcal{V}_r \subseteq \mathcal{B} \setminus (\mathcal{V}_b^* \cap \mathcal{P})$. By Theorem 2 it only remains to verify that $\mathcal{V}_r$ lies in a single connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{P})$, and to actually compute the cast parts.

We compute a representation of $\mathcal{V}_b^* \cup \mathcal{P}$, and find all facets f of $\mathcal{P}$ with a downward normal. Using a linear-time graph transversal, we verify that all these facets lie in the same connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{P})$. If so, this will be the blue cast part $\mathcal{C}_b$.

Since every connected component of $\mathcal{V}_r$ must be bounded by some facet f with a downward normal, this computation indeed establishes that $\mathcal{V}_r$ lies in $\mathcal{C}_b$.

Theorem 10 *Given a pair of directions, we can determine the castability of a simple polyhedron with n vertices in $O(n^2\log^2 n)$ time and $O(n^2\log n)$ space. If castable, the cast parts can be constructed in the same time and space bounds. The cast parts constructed have $O(n^2)$ complexity, which is asymptotically optimal.*

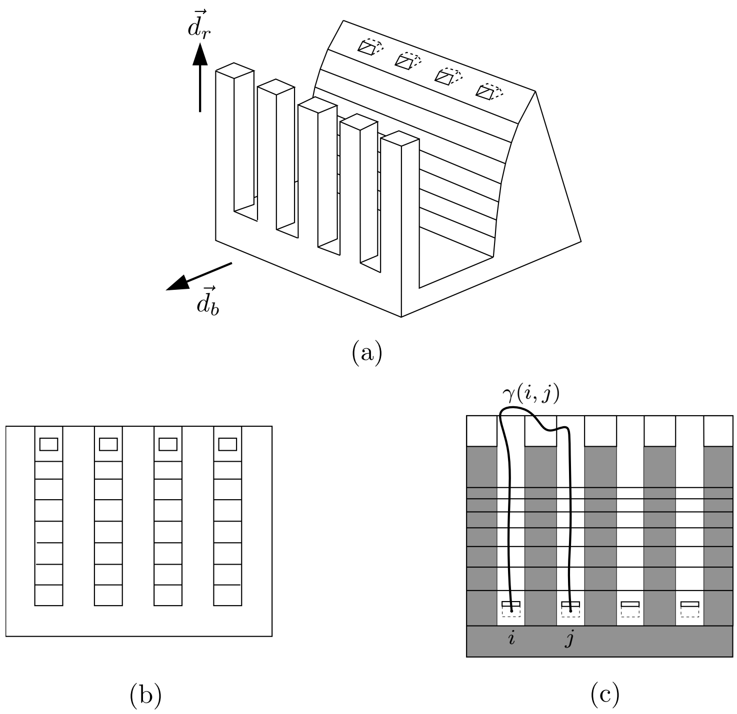


Figure 9: (a) A polyhedron with five vertical legs and four small holes, (b) The visibility map from $\vec{d}_b$, and (c) The visibility map from $\vec{d}_r$

Lemma 6 implies that $\mathcal{V}_b^*$ has $O(n^2)$ complexity. So the cast parts have $O(n^2)$ complexity. The time and space complexities of the construction follow from Lemma 9 and the linear-time graph traversal.

The optimality of the complexity of our cast parts follows from the example in Figure 9. The polyhedron shown in Figure 9(a) has five vertical "legs", and four small holes such that parts of the cast inside the holes can only be removed in direction $\vec{d}_b$. The blue shadow $\mathcal{S}_b$ has $\Theta(n^2)$ complexity (Figure 9(b) shows the visibility map as seen in direction $-\vec{d}_b$). It follows that the lower envelope $\mathcal{L}$ of all lower blue curtains and blue shadow facets has complexity $\Omega(n^2)$. Figure 9(c) shows the blue shadow of the object. This shows that the analysis of the cast part size of our construction is tight. Since the holes in the object are in the red shadow volume, they need to be in $\mathcal{C}_b$. The blue shadow on $\mathrm{bd}(\mathcal{P})$, on the other hand, must belong to $\mathcal{C}_r$. In any cast construction, there must be a path $\gamma(i,j)$ connecting two points i and j in two holes through $\mathrm{int}(\mathcal{C}_b)$. Let $\gamma^*(i,j)$ be the projection of the path on $\mathrm{bd}(\mathcal{P})$ in $-\vec{d}_r$. We claim that 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. Figure 9(c) shows a path $\gamma(i,j)$ connecting points i and j in two holes. 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.

4 Finding a pair of directions

We have seen how to test whether $\mathcal{P}$ 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: 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 [14]. 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. 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 11 *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 vertices of $\mathcal{P}$ lie in a plane parallel to the plane determined by $\vec{d}_r$ and $\vec{d}_b$.*

Proof. The castability of $\mathcal{P}$ depends on three factors:

- $\mathcal{S}_r \cap \mathcal{S}_b = \emptyset$.
- $\mathcal{V}_r \cap \mathcal{V}_b = \emptyset$. This is equivalent to the condition of Lemma 8.
- $\mathcal{V}_r$ lies in one connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{P})$.

Let's first consider the first condition: $\mathcal{S}_r \cap \mathcal{S}_b = \emptyset$ if and only if $\mathcal{S}_b$ is completely visible from the red direction. The blue shadow $\mathcal{S}_b$ 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). Similarly, 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 $\mathcal{S}_b$. Again this is included in case (ii).

Consider now the arrangement of all blue curtains. It can change combinatorially only 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 vertex of $\mathcal{P}$ passes over an edge of $\mathcal{P}$ (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 $\mathcal{P}$ 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 red and blue shadow and of the projection of the curtains cannot change. Neither can the combinatorial structure of the arrangement of the blue curtains. It follows that the combinatorial structure of $\mathcal{V}_b^*$ cannot change without leading to a configuration as postulated in the lemma. Without such a change, $\mathcal{V}_r$ cannot change from lying in one connected component of $\mathcal{B} \setminus (\mathcal{V}_b^* \cup \mathcal{P})$. □

We can now turn this characterization into an algorithm.

Theorem 12 *Given a simple polyhedron with* n *vertices, we can in time* $O(n^{14} \log^2 n)$ *construct a set of all possible pairs of directions in which the polyhedron is castable.*

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

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 $O(n^2 \log^2 n)$ time, we conclude that all directions for which there is a good cast can be computed in $O(n^{14} \log^2 n)$ time. □

References

[1] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22:794–806, 1993.

[2] H.-K. Ahn, M. de Berg, P. Bose, S.-W. Cheng, D. Halperin, J. Matoušek, and O. Schwarzkopf. Separating an object from its cast. *Computer-Aided Design*, 34:547–559, 2002.

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

[4] 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.

[5] P. Bose, D. Bremner, and M. van Kreveld. Determining the castability of simple polyhedra. *Algorithmica*, 19:84–113, September 1997.

[6] P. Bose, M. van Kreveld, and G. Toussaint. Filling polyhedral molds. *Comput. Aided Design*, 30:245–254, April 1998.

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

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

[9] R. Elliot. *Cast Iron Technology.* Butterworths, London, 1988.

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

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

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

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

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

[15] W. Pribble. Molds for reaction injection, structural foam and expandable styrene molding. In J. DuBois and W. Pribble, editors, *Plastics Mold Engineering Handbook.* Van Nostrand Reinhold Company, New York, 1987.

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

[17] C. Walton and T. Opar, editors. *Iron Castings Handbook.* Iron casting society, Inc., 1981.