On the Separation of a Polyhedron from Its Single-Part Mold*

Dan Halperin† Shahar Shamai†

Abstract

Casting is a manufacturing process where liquid material is poured into a mold having the shape of a desired product. After the material solidifies, the product is pulled out of the mold. We study the case in which the mold is made of a single part and the object to be produced is a three-dimensional polyhedron. We give an algorithm that decides whether a given polyhedron with n facets can indeed be produced that way, and if so indicates how to orient the polyhedron in the mold and in what directions can the product be pulled out without breaking the mold. Our algorithm runs in $O(n \log n)$ time. The best previous algorithm for this problem that we are aware of runs in $O(n^2)$ time. For a convex polyhedron we present a linear-time algorithm.

1 Introduction

Casting is a widely-used manufacturing process, where liquid material is poured into a cavity inside a mold, which has the shape of a desired product. After the material solidifies, the product is taken out of the mold. Typically a mold is used to manufacture numerous copies of a product, in which case we need to make sure that the solidified product can be separated from its mold without breaking it.

The problems that we study here belongs to the larger topic termed *Movable Separability of Sets* by Toussaint [7]. Problems in this area are often exceedingly challenging from a combinatorial- and computational-geometry point of view (see, e.g., [6]). At the same time solutions to these problems are needed in mold design [1], assembly planning [5], and 3D printing to mention a few application areas.

We focus in this paper on a fairly basic movable-separability question. We are given a polyhedron P in $\mathcal{R}^3$ with n facets. We do not make any particular assumptions about the polyhedron besides that it is a closed regular set, namely it does not have dangling edges or facets. The mold is box-shaped and the cavity has the shape of P such that one of P's facets is the top facet of the cavity. See Figure 1 for an illustration in 2D. Once the top facet has been determined, we wish to detect whether there is a direction in which the solidified object could be pulled out of the mold, namely to find a direction that has a positive component in the positive z-direction and such that P could be pulled out of the mold without colliding into the mold. If a direction is found we say that the corresponding top facet is *valid*, and that the polyhedron P is *castable.*

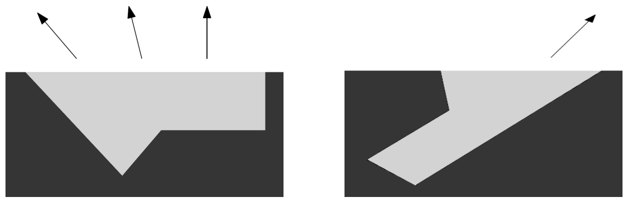

Figure 1: Polygons (light grey) in their molds (darker grey) and valid pull-out directions.

We address two problems:

All Facets Single Direction (AFSD): Determine which facets of P can serve as a valid top facet and for each such facet indicate *one* direction in which P can be pulled out of the mold.

All Facets All Directions (AFAD): Same as above but for each valid facet indicate *all* the directions in which P can be pulled out of the mold.

Why would anyone bother to solve AFAD and not suffice with AFSD? There could be advantages to computing all possible directions. First, it is a more stable solution if there is a continuum of directions rather than a single direction of separation. Second, we could use the availability of many possible directions to impose some direction-related optimality criteria.

Contribution The current algorithms that we are aware of [2, Chapter 4] solve AFSD in $O(n^2)$, and imply an $O(n^2 \log n)$ time solution for AFAD. Both solutions rely on handling each candidate facet to be a top facet separately. Both algorithms, as well as the algorithms that we present below use linear storage space. Our contribution in this paper is an $O(n \log n)$-time algorithm for the AFAD problem. We also present an

*This work has been supported in part by the Israel Science Foundation (grant no. 1102/11), by the German-Israeli Foundation (grant no. 1150-82.6/2011), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

†The Blavatnik School of Computer Science, Tel Aviv University, danha@post.tau.ac.il, shasha94@gmail.com.

$O(n)$-time solution to the AFAD problem when the input polyhedron is restricted to be convex.

2 One Structure for All Facets

Instead of handling each candidate top facet separately, as in [2, Chapter 4], we handle all candidate top facets simultaneously using an arrangement of great circles on the unit sphere $\mathcal{S}^2$. Each point p on $\mathcal{S}^2$ represent a direction in $\mathcal{R}^3$—the direction of the vector from the center of $\mathcal{S}^2$ to p. We will use the terms *points* and *directions* on $\mathcal{S}^2$ interchangeably.

Let $F_1, \ldots, F_n$ be the facets of the given polyhedron P. Let $\nu(F_i)$ be the normal to the facet F_i pointing into the polyhedron.

A valid mold is described by F_i as its top facet and a valid pull-out direction $\vec{d'}$. Since we wish to argue about all candidate top facets simultaneously, we will describe a mold by the pair $(F_i, \vec{d})$ which should be interpreted as follows. The top facet of the mold is F_i. To achieve this, the polyhedron needs to be rotated such that F_i becomes the top facet. We apply the same rotation matrix R_i to $\vec{d}$ to obtain a pull-out direction $\vec{d'} := R_i\vec{d}$.

Lemma 1 *The pair $(F_i, \vec{d})$ represents a valid mold and pull-out direction if and only if (i) $\vec{d} \cdot \nu(F_i) < 0$ and (ii) $\forall j \neq i,\ \ \vec{d} \cdot \nu(F_j) \geq 0$.*

Proof. For the case where F_i is the top facet of P, this fact is proved in Lemma 4.1 in [2]. It remains to notice that the Conditions (i) and (ii) are invariant under rotation. They hold in P's given orientation if and only if they hold when P is rotated such that F_i becomes the top facet. $\square$

Henceforth, we will call a pair $(F_i, \vec{d})$ that obeys the conditions of Lemma 1 a *valid pair*. Notice that the actual pull-out direction is $R_i\vec{d}$, where R_i is the matrix that rotates P such that F_i becomes the top facet, or equivalently such that $\nu(F_i)$ points vertically downwards.

Denote by h_i the closed hemisphere of directions $\vec{d}$ on $\mathcal{S}^2$ for which $\vec{d} \cdot \nu(F_i) \geq 0$, and by $\bar{h}_i$ the open complement hemisphere. Let $\bar{H} = \{\bar{h}_1, \ldots, \bar{h}_n\}$. Let c_i denote the boundary great circle of h_i, and let $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$. Consider the arrangement $\mathcal{A}(\mathcal{C})$ on $\mathcal{S}^2$, namely the subdivision of $\mathcal{S}^2$ induced by the great circles in $\mathcal{C}$ into cells of dimensions $0, 1$ and 2, which we refer to as vertices, edges, and faces respectively.

Definition 1 *The* depth *of a point p on $\mathcal{S}^2$ is the number of hemispheres in $\bar{H}$ in which p is contained.*

Observation 1 *All the points in any fixed cell of the arrangement $\mathcal{A}(\mathcal{C})$ have the same depth.*

The key observation leading to our efficient solution is expressed in the following theorem.

Theorem 2 *The polyhedron P is castable with a single-part mold if and only if the arrangement $\mathcal{A}(\mathcal{C})$ contains a point of depth 1. A cell ξ of depth 1 in $\mathcal{A}(\mathcal{C})$, which is covered by the hemisphere $\bar{h}_i$, represents a mold whose top facet is F_i and each point $\vec{d}$ in ξ represents the valid pull-out direction $R_i\vec{d}$, where R_i is the orthonormal matrix that rotates $\nu(F_i)$ to point vertically down (in the negative z direction).*

Proof. Let ξ be a cell of depth 1 in $\mathcal{A}(\mathcal{C})$ covered by $\bar{h}_i$, and let $\vec{d}$ be a point in ξ. We establish that the pair $(F_i, \vec{d})$ is a valid pair by verifying that the conditions of Lemma 1 above hold for it.

It remains to show that no point in any cell of different depth can represent a valid pull-out direction for any top facet. Consider a cell ψ of depth greater than 1 and a point $\vec{d}$ in it. Let J be the index set of the hemispheres $\bar{h}_i$ that cover ψ: $J = \{i | \vec{d} \in \bar{h}_i\}$. One of the facets $F_j, j \in J$ must serve as the top facet for Condition (i) of Lemma 1 to hold. But then for each of the remaining facets $F_k, k \in J, k \neq j$ Condition (ii) of the lemma is violated. Finally, if a cell has zero depth, then no facet can serve as top facet for any pull-out direction described by points in this cell. $\square$

Remark. Notice that in our setting there cannot be a face of zero depth in $\mathcal{A}(\mathcal{C})$. Similarly, in Lemma 1, Condition (i) follows from Condition (ii).

Our goal is then to compute the cells of $\mathcal{A}(\mathcal{C})$ of depth 1. We first investigate the overall complexity of all these cells.

Proposition 3 *The overall complexity of the depth-1 cells in $\mathcal{A}(\mathcal{C})$ is $O(n)$.*

Proof. We refer to the horizontal great circle of $\mathcal{S}^2$ as the *equator*. The equator splits $\mathcal{S}^2$ into the open lower hemisphere and the open upper hemisphere. We will handle each of these three parts of $\mathcal{S}^2$ separately.

Assume temporarily that no hemisphere h_i has the equator as it bounding great circle.

We start with the upper hemisphere of $\mathcal{S}^2$. Centrally project the intersection of the upper hemisphere with each of the hemispheres $\bar{h}_i$ onto the plane $z = 1$. For each $\bar{h}_i$ we obtain a halfplane $\bar{g}_i$. Let g_i denote the complementary closed half plane, and ℓ_i the line bounding each of them. Let G, $\bar{G}$, and $\mathcal{L}$ be the corresponding sets of n elements each. See Figure 2 for an illustration.

We call a vertex v of the arrangements $\mathcal{A}(\mathcal{L})$, which is the intersection point of two distinct lines in $\mathcal{L}$, a *configuration*. We say that such a vertex v is in conflict with the halfplane $\bar{g} \in \bar{G}$ if $v \in \bar{g}$. We are interested in counting all the vertices of $\mathcal{A}(\mathcal{L})$ that have

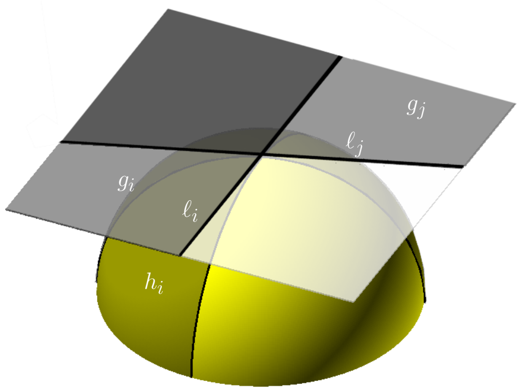

Figure 2: Central projection of hemispheres onto the plane $z = 1$.

exactly one conflict. These will correspond to vertices of depth 1 in the upper hemisphere of $\mathcal{A}(\mathcal{C})$; more generally the number of conflicts of a configuration is the depth of the vertex. Let $N_k(\mathcal{L})$ denote the number of vertices of depth k, and $N_{\leq k}(\mathcal{L})$ denote the number of vertices of depth at most k. Let $N_k(n)$ (resp. $N_{\leq k}(n)$) denote the maximum $N_k(\mathcal{L})$ (resp. $N_{\leq k}(\mathcal{L})$) over all sets $\mathcal{L}$ of n lines. To bound the maximum number of vertices of depth 1 we use the Clarkson-Shor framework [3], which asserts that

$$N_{\leq k}(n) = O(k^d N_0(n/k)) \ ,$$

where d is the number of objects in the set that define a configuration. In our case $d = 2$ and $k = 1$.

What we still need to bound is $N_0(n)$, namely the maximum number of configurations with no conflicts in any set $\mathcal{L}$ of n lines. However, this is easy: these are the vertices on the boundary of the intersection of the halfplanes in G. Therefore $N_0(n) = n$. It follows that $N_{\leq 1} = O(n)$ as asserted.

Analogous arguments apply to the lower hemisphere.

Next, we consider $\mathcal{A}(\mathcal{L})$ restricted to the equator as a one-dimensional arrangement consisting of vertices and arcs, namely 0- and 1-dimensional cells. It is not difficult to see that the complexity of the cells of depth 1 in this arrangement is $O(1)$. There can be at most four such cells—in a degenerate situation where there are two pairs of complementary half-circles, which in turn induce two pairs of antipodal 0-cells of depth 1.

We now relax the assumption that there are no hemispheres whose bounding circle is the equator. Notice that the introduction of such hemispheres does not affect the asymptotic upper bounds derived so far. There is the possible introduction of a constant number of extra vertices of depth 1 on the equator. $\square$

Can one compute these faces efficiently? The Clarkson-Shor framework [3] also gives a randomized algorithm to compute these faces in expected near-linear time. However, we will next describe a simple deterministic algorithm running in $O(n \log n)$ time for that purpose.

3 The Algorithm

We now compute all the valid pairs $(F_i, \vec{d})$, by computing all the cells of depth 1 in the arrangement $\mathcal{A}(\mathcal{C})$. We handle each of the upper hemisphere, the equator, and the lower hemisphere separately.

For the upper hemisphere we use the projection described above, the set $\bar{G}$ of half-planes, and the set $\mathcal{L}$ of their bounding lines. Notice that there is one-to-one correspondence between the points on the plane $z = 1$ and the points of the open upper hemisphere. Thus once we find the cells of depth 1 in $\mathcal{A}(\mathcal{L})$ we immediately get the desired pairs. Hence we focus on the plane $z = 1$.

We subdivide the set $\bar{G}$ into the disjoint union of three subsets: $\bar{G}_1, \bar{G}_2$, and $\bar{G}_3$, having their half-planes above their bounding line, below their bounding line, or having a vertical bounding line, respectively. Let $\mathcal{L}_i$ be the set of bounding lines of $\bar{G}_i, i = 1, 2, 3$. Let λ_i be the number of lines in $\mathcal{L}_i$.

Recall the definition of levels in arrangements, which we will need below. Given a set $\mathcal{L}$ of lines in the plane, we define the *level* of a point p in the plane to be the number of lines in $\mathcal{L}$ strictly below p [4]. We say that an edge e in the arrangement $\mathcal{A}(\mathcal{L})$ is at level k if there is a point in the interior of e at level k (and hence all interior points of e are at level k). The k-level of the arrangement $\mathcal{A}(\mathcal{L})$ is the closure of the union of edges of $\mathcal{A}(\mathcal{L})$ that are at level k. See Figure 3.

Back to the depth-1 faces, we handle the three sets separately and then merge the results. We start with $\bar{G}_1$. If we restrict our attention to the half-planes in $\bar{G}_1$, the points of depth k in the plane correspond exactly to points in the plane at level k. Thus our goal is to compute the points at levels 0 and 1. (We need level 0 as well since we will later merge this result with the results for the other sets.) We use divide-and-conquer on the lines in $\mathcal{L}_1$. Assume that we have already computed the points at levels 0 and 1 for each of the arrangements $\mathcal{A}(L_1^a)$ and $\mathcal{A}(L_1^b)$, where $\mathcal{L}_1^a$ and $\mathcal{L}_1^b$ are two disjoint subsets of $\mathcal{L}_1$ of size $\lambda_1/2$ each. The complexity of level 0 in each is obviously $O(\lambda_1)$ and so is the complexity of level 1 (this is well known; it follows for example as a special case of Proposition 3 above). What we actually compute are the levels 0 and 1 of the arrangements, namely two polygonal lines rather than a planar subdivision. (This will no longer be true when we merge the final result for $\bar{G}_1$ with the results for the other subsets of $\bar{G}$.) Therefore we can easily carry out the current merge step by projecting the breakpoints of both levels in each arrangement onto the x-axis and merging these lists of projected points into one list. We then handle in constant time the slab above each interval between two consecutive breakpoint projections. This way we obtain the level 0 and level 1 faces of the arrangement in time $O(\lambda_1 \log \lambda_1)$.

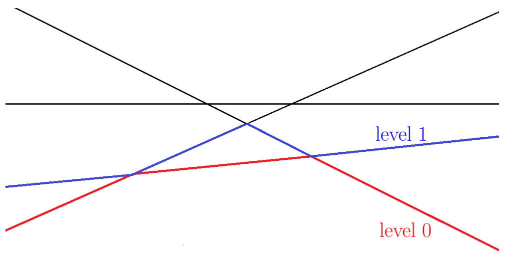

Figure 3: Levels in an arrangement of lines.

We operate analogously on the set $\bar{G}_2$, and merge the results with those already computed for $\bar{G}_1$ in a merge step that is similar to the merge step we have just described for $\bar{G}_1$, still keeping the faces of levels 0 and 1. Now, however, we need to maintain a planar subdivision, say by using a DCEL [2, Chapter 2]. From this point on, we also record with each cell of depth 1 which is the half-plane that covers it.

Next we compute the relevant faces for $\bar{G}_3$. This is easier and can be carried out in $O(\lambda_3)$ time , resulting in only a constant number of faces. We omit the details here. We finally merge this constant size result with the result of the preceding step in $O(n)$ time. This concludes the handling of the upper hemisphere of $\mathcal{S}^2$.

The lower hemisphere is handled analogously. Computing the depth-1 cells on the equator is simpler yet and can be carried out in $O(n)$ time by an incremental algorithm adding the circular arcs describing forbidden directions one after the other. At any time during the algorithm there is a constant number of arcs of depth 0 and 1.

At all times our algorithm does not require more than $O(n)$ storage as we keep a constant number of linear lists (describing levels 0 and 1 of certain arrangements), or a planar subdivision of linear size. In summary

Theorem 4 *Given a polyhedron P with n facets, we can find in $O(n \log n)$ time all the valid pairs $(F_i, \vec{d})$, where P can be pulled in direction $R_i\vec{d}$ out of a mold having F_i as a top facet, and R_i is the matrix that rotates P such that F_i becomes the top facet. The algorithm requires linear space.*

Finally we state our more efficient algorithm for the analogous AFAD problem for an arbitrary polygon in the plane. The proof is omitted for lack of space.

Theorem 5 *Given a polygon Q with n edges in the plane, we can find in $O(n)$ time all the valid pairs of top edge and pull-out directions. The algorithm uses only constant-size working storage.*

4 Casting Convex Polyhedra

In this section we show how to determine the castability of a *convex* polyhedron more efficiently, still solving the AFAD problem for this case.

We say that two facets of the input polyhedron P are neighbors if their closures intersect in an edge. Denote by $\mathcal{M}_i$ the set of neighbors of the facet F_i, by m_i the cardinality of this set, and by J_i the index set of the facets in $\mathcal{M}_i$. The efficient algorithm stems from the following observation (proof omitted).

Proposition 6 *For a convex polyhedron, the pair $(F_i, \vec{d})$ represents a valid mold and pull-out direction iff (i) $\vec{d} \cdot \nu(F_i) < 0$ and (ii) $\forall j \in J_i, \vec{d} \cdot \nu(F_j) \geq 0$.*

The algorithm proceeds by fixing a face F_i, computing its edge-neighboring facets and computing the intersection of allowable directions (half-planes on $z = 1$) in $O(m_i)$ time, using the order of the neighbors along the boundary of f. We repeat the procedure for each facet of P. Notice that m_i is in fact the number of edges on the boundary of F_i. The overall cost $O(m_i)$ over all candidate top facets F_i is $O(n)$ by Euler's formula. In summary

Theorem 7 *The AFAD problem for a* convex *polyhedron P with n facets is solvable in time $O(n)$.*

References

[1] H. Ahn, M. de Berg, P. Bose, S. Cheng, D. Halperin, J. Matoušek, and O. Schwarzkopf. Separating an object from its cast. *Computer-Aided Design*, 34(8):547–559, 2002.

[2] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd edition, 2008.

[3] K. L. Clarkson and P. W. Shor. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

[4] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 529–562. Chapman & Hall/CRC, 2nd edition, 2004.

[5] D. Halperin, J. Latombe, and R. H. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 26(3-4):577–601, 2000.

[6] J. Snoeyink and J. Stolfi. Objects that cannot be taken apart with two hands. *Discrete & Computational Geometry*, 12:367–384, 1994.

[7] G. Toussaint. Movable separability of sets. In *Computational Geometry*. North-Holland Publishing Company, 1985.