Separating an Object from its Cast

Hee-Kap Ahn[1] Mark de Berg[2] Prosenjit Bose[3] Siu-Wing Cheng[4]

Dan Halperin[5] Jiří Matoušek[6] Otfried Schwarzkopf[1]

Abstract

We study geometric problems that arise in *casting*. Casting is a manufacturing process where 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. In this paper we consider the case where the cast consists of two parts and we address problems concerning the removal of the cast parts. In particular, we study the following question: Given a cast for a object and a direction $\vec{d}$, is it possible to partition the cast into two parts in such a way that the parts can be removed in directions $\vec{d}$ and $-\vec{d}$, respectively, without colliding with the object or the other cast part? The second question we study is: Given a cast for an object, is there a direction $\vec{d}$ that allows a partitioning of the cast into two parts that can be removed without collision into directions $\vec{d}$ and $-\vec{d}$, respectively? We give necessary and sufficient conditions for both cases, and we present algorithms to decide these questions for polyhedral objects. We also present a surprising result on the case where the cast parts do not have to be removed in opposite directions.

[1]Department of Computer Science, Pohang University of Science and Technology, Hyoja-Dong, Pohang 790-784, South Korea. Supported by the non-directed research fund of the Korean Ministry of Education, and partially by Postech Grant 96F004.

[2]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*).

[3]Département de Mathématiques et d'Informatique, Université du Québec à Trois-Rivières, C.P. 500, Trois-Rivieres, Québec G9A 5H7, Canada. Research supported by NSERC Grant OGP0183877 and a FIR Grant.

[4]Department of Computer Science, HKUST, Clear Water Bay, Hong Kong. Supported partly by the Research Grant Council CERG HKUST 650/93E and HKUST 650/95E.

[5]Department of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. Research supported by an Alon Fellowship, and by ESPRIT IV LTR Project No. 21957 (CGAL).

[6]Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic.

1 Introduction

Background. Computer-aided design systems have considerably simplified industrial design, the first phase in the life of a new product. To facilitate manufacturing and assembly, it is desirable to design parts in such a way that manufacturing and assembly can be performed easily, and therefore cheaply (*design for assembly*, *design for manufacturing*). To make this feasible, computer-aided design systems have to be augmented with a component that verifies that a part being designed can actually be manufactured using the intended techniques. Algorithms are required that can deduce the feasibility of manufacturing and assembly purely on the basis of a CAD model of the part.

Such algorithms have been proposed for a number of manufacturing processes, such as injection molding [4, 8, 21], NC-machining [11], and stereolithography [1]. For an overview of this application area, the reader is referred to the survey by Bose and Toussaint [2, 5].

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 [24], part orienting [9], fixturing [18], and welding [17], are just a few of many examples.

Although these manufacturing and assembly processes may be totally different from one another, some of them raise similar geometric problems that can be generally termed *separability* problems [22]. This is the case in the manufacturing process that we study in this paper: the process of casting [7, 23].

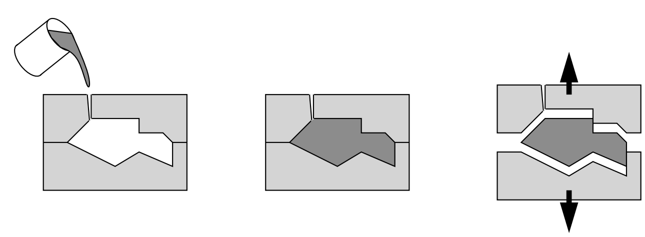

Figure 1: The casting process.

The casting process consists of filling with some molten liquid the cavity between two cast parts. After the liquid has solidified, the cast parts are removed and one is left with an object whose shape is that of the cavity. Figure 1 illustrates the process on a 2-dimensional example. The key property necessary for casting is that the cast parts must 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. Observe that the cast parts in the example are removed in opposite directions. This need not always be possible; sometimes it may be necessary to remove them in non-opposite directions.

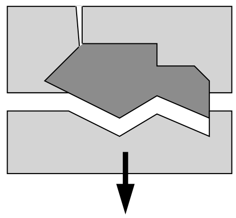

Figure 2: The top part of the cast is stuck.

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 here 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*.

We note that more complicated objects may be constructed through the use of cores and inserts [7, 20, 23]. As their names suggest, cores and inserts are appendages to the cast parts. These appendages are removed after the liquid has hardened and before the cast parts themselves are removed. They allow the possibility of building more complicated objects, in the sense that if the cores and inserts 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. Thus, it is preferable to avoid the use of cores and inserts if possible. 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 [21]. They present 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 [13] gave a heuristic approach to the 3-dimensional problem. They sample a set of points on the boundary of the given object and choose a set of candidate parting directions ordered by a heuristical function. Each candidate parting direction is checked in order and for each direction, the sampled points are tested if they are obscured by other parts of the product. If not, that parting direction is assumed to be feasible.

Kwong [14] gave an algorithm that can determine if a given parting direction is feasible for opposite cast-removal. The key idea is that if all the facets can be completely illuminated from the parting direction and its opposite, then the parting direction is feasible. The problem is thus reduced to the hidden surface removal problem in computer graphics.

Chen et al. [6] observe that any parting direction is feasible for a convex polyhedron. They therefore first compute the convex hull of a polyhedral object, and then obtain the *pockets* of the object by subtracting it from its convex hull. They observe that if all pockets are completely visible in either the parting direction or its opposite, then the parting direction is feasible, and give an algorithm that returns the parting direction that maximizes the number of completely visible pockets. However, as the converse of the above observation is not necessarily true, their algorithm is not complete and is not guaranteed to find a parting direction even though one exists.

Hui [12] builds on the previous paper [6] and obtain exponential time non-complete algorithms that also take into account cores and inserts.

Finally, Bose et al. [3] considered a special model of casting, the *sand casting model*, where two parts of the cast must be separated by a plane. Given a simple polyhedron with n vertices, they present an $O(n^{3/2+\varepsilon})$ time and space algorithm[1] (based on partition trees [16]), for any fixed $\varepsilon > 0$, as well as an $O(n^2)$ time and $O(n)$ space algorithm (based on linear programming) that decides whether there is a cast whose constituent parts can be removed in opposite directions. When arbitrary directions are allowed, their partition tree based algorithm still takes $O(n^{3/2+\varepsilon})$ time and space, however their linear programming based algorithm increases slightly to $O(n^2 \log n)$ time and $O(n)$ space. Note that even convex polyhedra are not castable in general if the two cast parts must be separated by a plane [3].

Summary of results. In this paper we give a complete characterization of castability for general objects, and obtain an algorithm to verify this condition for polyhedral objects. We also present an algorithm that reports all *combinatorially distinct directions* in which there is a good cast (we postpone a formal definition of "distinct directions" to Section 4). We do not assume any special separability of the two cast parts, and, generalizing previous work, allow parts of arbitrary genus. The running time of our algorithm for determining the legal parting directions is $O(n^4)$, where n is the combinatorial complexity of the polyhedral model. We show that there exist polyhedra for which there are $\Omega(n^4)$ combinatorially distinct directions in which there is a good cast. This implies that our algorithm is optimal in the worst case if we want to report all such directions.

In sand casting, the problem of finding a good cast reduces to the problem of finding a plane subdividing the boundary of the part into two terrains. Extending the boundary subdivision into two cast parts is trivial: one simply splits the cast with this plane. In our more general setting, the object boundary must also be split into terrains, but extension of the boundary subdivision to a legal cast is no longer straightforward. 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. This implies that Hui and Tan's [13] and Kwong's [14] approaches are not sufficient for finding a legal cast with arbitrary removal directions. (For opposite cast removal we prove that such an extension is always possible.) 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, for a number of reasons, and we therefore do not study this case in more detail.

Outline. The rest of the paper is organized as follows. In the next section we introduce more terminology and certain assumptions on the objects. In Section 3 we present criteria for testing whether a given object admits opposite cast removal in a given direction $\vec{d}$, and give an algorithm to test this condition for polyhedral objects. In Section 4 we give an algorithm to compute all possible directions of opposite cast removal for a given polyhedron. Experimental results are presented and discussed in Section 5. The case of non-opposite

[1] Bose et al. remark that this can be improved to $O(n^{4/3+\varepsilon})$.

cast removal is briefly addressed in section 6. We conclude in Section 7 with some open problems.

2 Preliminaries: Assumptions and Terminology

Throughout this paper, $\mathcal{P}$ denotes a polyhedron. We denote the boundary of $\mathcal{P}$ by $\partial\mathcal{P}$, namely $\partial\mathcal{P}$ denotes the union of vertices, edges, and facets on the boundary of $\mathcal{P}$. We assume that $\partial\mathcal{P}$ is a connected 2-manifold. Each facet of $\mathcal{P}$ is a connected planar polygon which is allowed to have polygonal holes. We assume that two adjacent facets cannot be coplanar, but allow coplanar non-adjacent facets. We also assume that $\mathcal{P}$ is *simple*, namely there is no pair of non-adjacent facets sharing 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 [19].

For generality—and since we are aware that many industrial parts are not polyhedral—we characterize castability for more general objects. We assume that an object $\mathcal{B}$ has a piecewise smooth boundary that is a 2-manifold of arbitrary genus. $\mathcal{B}$ has to be topologically equivalent to a polyhedron $\mathcal{P}$ as defined above. 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.

We will assume that the outer shape of the cast is a rectangular block. A cast, denoted by $\mathcal{C}$, consists of a *red part* $\mathcal{C}_r$ and a *blue part* $\mathcal{C}_b$. The union of $\mathcal{B}$, $\mathcal{C}_r$ and $\mathcal{C}_b$ is assumed to be an axis-parallel box.

In our discussion we will refer to the subdivision of the unit sphere $\mathcal{S}^2$ 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 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 face 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.

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 it is possible to build a cast $\mathcal{C} = (\mathcal{C}_r, \mathcal{C}_b)$ for $\mathcal{B}$ such that the red part $\mathcal{C}_r$ of the cast can be translated to infinity in direction $\vec{d}$ without colliding with $\mathcal{P}$ or $\mathcal{C}_b$, and $\mathcal{C}_b$ can be translated in direction $-\vec{d}$ without collision. We say that $\mathcal{B}$ *can be cast* in direction $\vec{d}$ if this is the case. The order of removing the cast parts is irrelevant in this situation.

Throughout this section we assume that $\vec{d}$ is the vertical direction (the positive z direction). 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. Suppose that a vertical line intersects the interior of $\mathcal{B}$ in more than one connected component. Between those connected components there must be a point p that belongs to either $\mathcal{C}_r$ or $\mathcal{C}_b$. However, p can be translated neither upward nor downward without colliding with $\mathcal{B}$, a contradiction.

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 now $\mathcal{B}^*$ be the solid obtained by sweeping $\mathcal{B}$ upward to infinity. By the assumption, $\mathcal{B}^* \setminus \mathcal{B}$ can be moved upward without colliding with $\mathcal{B}$. We can therefore set $\mathcal{C}_r := B \cap (\mathcal{B}^* \setminus \mathcal{B})$ and $\mathcal{C}_b := B \setminus (\mathcal{B} \cup \mathcal{C}_r)$. Clearly, the red cast part $\mathcal{C}_r$ can be moved upward without collision. Let now q be a point in $\mathcal{C}_b$, and consider the vertical ray emanating downward from q. This ray cannot contain a point of $\mathcal{C}_r$, and it cannot intersect the interior of $\mathcal{B}$. Therefore, $\mathcal{C}_b$ can be moved downward to infinity. $\square$

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 and a blue part. We call a facet of $\mathcal{P}$ an *up-facet* if its outward normal points upward, and a *down-facet* if its outward normal points downward. 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 and 4 illustrates that vertical facets sometimes need to receive both colors. The facet $abcd$ is coplanar with the vertical facets incident to the edges uv and rs. The construction in the proof clearly colors all up-facets red and all down-facets blue. The vertical facets are all colored blue as well, except for the facet $abcd$ which is subdivided by pq with $abpq$ colored red and $pqcd$ colored blue. In fact, to obtain a cast for this object in the vertical direction, the facet $abcd$ has to receive two different colors.

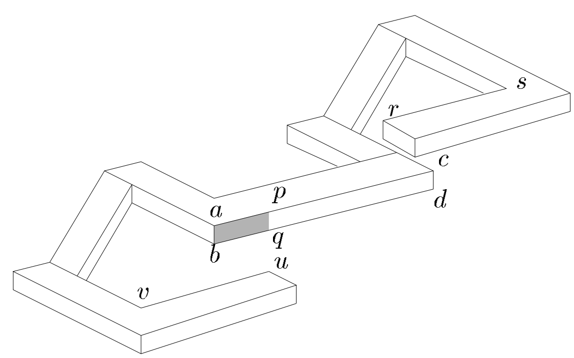

Figure 3: The facet $abcd$ is coplanar with the vertical facets incident to uv and rs. The line segment pq is the intersection of $abcd$ with a ray shooting vertically upward from u.

The construction used in Lemma 3.1 does not result in practically useful casts, since it generates vertical walls between the red and blue cast part. Sometimes these are unavoidable (as for the object in Figure 3). However, we next present an alternative construction that does not create any vertical walls if they are not necessary.

Theorem 3.2 *We can construct a vertical cast for a vertically monotone polyhedron $\mathcal{P}$ with n vertices in time $O(n \log n)$. The two cast parts do not meet along vertical*

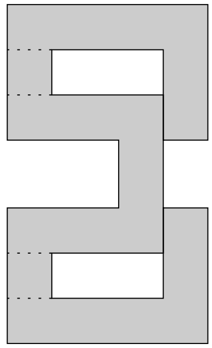

Figure 4: Projection onto the xy-plane of the object in Figure 3, with the interior shaded.

facets, if no vertical line avoiding the interior of $\mathcal{P}$ touches two non-adjacent facets of $\mathcal{P}$.

Proof: Let h be an arbitrary plane that is parallel to the xy-plane and cuts the box B into two halves, and let R be the rectangle $R := h \cap B$. We project $\mathcal{P}$ onto h and obtain a polygon $\overline{\mathcal{P}}$ with holes as in Figure 4. This can be done in time $O(n \log n)$ using a simple plane sweep algorithm.

Since $\mathcal{P}$ is vertically monotone, every point on $\partial\overline{\mathcal{P}}$ must be the projection of a point on $\partial\mathcal{P}$. Indeed, with each vertex $\overline{v}$ of $\overline{\mathcal{P}}$, we associate a vertex v of $\mathcal{P}$ that projects onto $\overline{v}$. If there is more than one such vertex, we choose the topmost one.

It follows that the complexity of $\overline{\mathcal{P}}$ is $O(n)$ and hence $\overline{\mathcal{P}}$ can be can be triangulated in $O(n)$ time, as shown in Figure 5. Every triangle is now "lifted" into three dimensional space by replacing every vertex $\overline{v}$ by its associated vertex v. We obtain a triangulated terrain σ inside the box B.

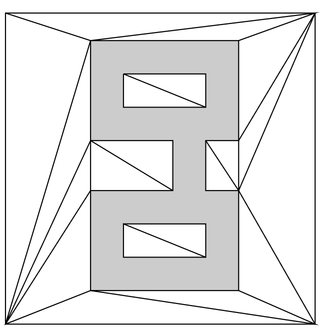

Figure 5:

Let now $\mathcal{U}$ be the union of all up-facets of $\mathcal{P}$ and σ, and let $\mathcal{U}^*$ be the solid obtained by sweeping $\mathcal{U}$ upward. We can now let $\mathcal{C}_r := B \cap \mathcal{U}^*$ be the red cast part. The blue cast part is defined as $\mathcal{C}_b := B \setminus (\mathcal{P} \cup \mathcal{C}_r)$.

By its definition, $\mathcal{C}_r$ can be moved upward without collision. Consider a point q in $\mathcal{C}_b$. Clearly, a vertical downward ray emanating from q cannot intersect $\mathcal{P}$. It also cannot intersect σ, and hence $\mathcal{C}_b$ can be moved downward to infinity.

It remains to argue that when no vertical line avoiding the interior of $\mathcal{P}$ touches two non-adjacent facets of $\mathcal{P}$, then the separation of $\mathcal{C}_r$ and $\mathcal{C}_b$ has no vertical facets. This follows from the fact that in that case $\mathcal{U}$ is a terrain without vertical facets that splits B into two parts. $\square$

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

Observation 3.3 *A polyhedron $\mathcal{P}$ is z-monotone if and only if the union of open up-facets forms a terrain.*

This observation is essentially basis for Kwong's algorithm [14], see the Introduction. We prefer a somewhat different approach, that decides monotonicity by looking at the silhouette edges of the polyhedron only. This is the key behind the efficient algorithm presented in the next section where we go over all possible directions and check in which of them the 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 can have vertical facets.

Let $\mathcal{B}$ be an object, and consider a vertical line ℓ. It 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 $\mathcal{P}$ is called a *convex silhouette interval*. A boundary interval that is surrounded on both sides by intervals in the interior of $\mathcal{P}$ is called a *reflex silhouette interval*. Over all vertical lines, the union of all convex silhouette intervals forms the *convex silhouette* and the union of all reflex silhouette intervals the *reflex silhouette*. The union of both 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 example with part of its quite complicated silhouette shown shaded. Note that the segments a and b are also part of the silhouette.

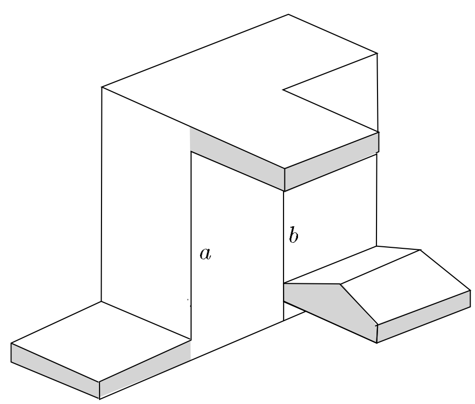

Figure 6:

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 convex silhouette (or short, the silhouette) of such an object consists of a finite number of disjoint curves or "bands" on $\partial\mathcal{P}$, the *silhouette curves*. Let $\pi(x)$ denote the vertical projection of x on the xy-plane. The silhouette curves project on closed curves in the plane. We will call these projected curves *shadow curves*. The key step in our argument is the following lemma.

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 mutually disjoint, simple, closed curves in the xy-plane. Hence we can express $\pi(S)$ as the disjoint union of topological circles $\gamma_1, \gamma_2, \ldots, \gamma_n$, which partition the xy-plane into open regions R_1, R_2, ..., R_k. Every region R_i is topologically equivalent to a disc with a finite number of holes, as in Figure 7.

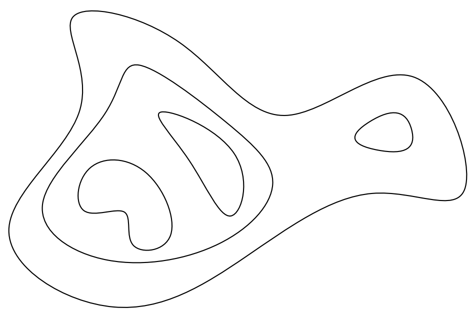

Figure 7:

For a point x in the xy-plane, $\pi^{-1}(x) \cap \mathcal{B}$ is the disjoint union of closed intervals. We number those intervals from bottom to top as $I_1(x)$, $I_2(x)$, ..., $I_m(x)$. Within each region R_i, the number m here is a constant.

Let now

$$P_{ij} := \bigcup_{x \in R_i} I_j(x),$$

and let $\overline{P}_{ij}$ denote the topological closure of P_{ij}. We observe that every $\overline{P}_{ij}$ is a closed, arc-connected set. Since the reflex silhouette of $\mathcal{B}$ is empty, for any i, j, and $j' \neq j$, $\overline{P}_{ij}$ and $\overline{P}_{ij'}$ are disjoint.

The difference $\overline{P_{ij}} \setminus P_{ij}$ consists of additional "cylinders" above the shadow curves bounding R_i. Consider now such a shadow curve γ_t between R_i and another region $R_{i'}$. Each piece of $\pi^{-1}(\gamma_t) \cap \mathcal{B}$ belongs to at most one $\overline{P}_{ij}$ and at most one $\overline{P}_{i'j'}$. It follows that every $\overline{P}_{ij}$ is adjacent to at most one $\overline{P}_{i'j'}$ in every $R_{i'}$ that is adjacent to R_i.

By the condition on $\pi(S)$, the adjacency relationship between the regions R_i has tree structure. This implies that there cannot be a path connecting $\overline{P}_{ij}$ to $\overline{P}_{ij'}$ for $j \neq j'$. Since $\mathcal{B}$ is arc-connected, this is only possible if there is at most one such P_{ij} above every region R_i, and hence $\mathcal{B}$ is z-monotone. ◻

The lemma does not apply to an object such as in Figure 3, since its shadow curve is not simple. To extend the lemma to the more general setting, we have to define clearly what kind of non-simplicity we allow. 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. We say that a set of shadow curves are *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.

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: Let S be the silhouette of $\mathcal{B}$. We slightly shrink $\mathcal{B}$ to obtain a new object $\mathcal{B}'$ by subtracting from $\mathcal{B}$ the Minkowski sum of $\partial\mathcal{B}$ and a ball of arbitrarily small radius $\varepsilon > 0$. Let S' be the silhouette of $\mathcal{B}'$. The silhouette moves slightly during the shrinking. It is not difficult to see that $\mathcal{B}'$ has empty reflex silhouette, and since $\pi(S')$ is obtained from $\pi(S)$ precisely as prescribed above in the definition of non-crossing, $\pi(S')$ consists of mutually disjoint, simple curves. It follows that no vertical line intersects more than one silhouette interval in $\mathcal{B}'$. Let ℓ be a vertical line stabbing the interior of a topological circle γ_t in $\pi(S)$. We can choose ε small enough such that ℓ stabs an open region inside γ'_t in $\pi(S')$. Lemma 3.5 implies that ℓ intersects the interior of $\mathcal{B}'$ in at most one connected component and hence the same holds for $\mathcal{B}$. ◻

To wrap up the argument, we need a final small lemma.

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 S be the silhouette of $\mathcal{B}$. We shrink $\mathcal{B}$ slightly to $\mathcal{B}'$ with silhouette S'. Let p be a point in $\pi(S)$ where a shadow curve meets itself or another shadow curve. Since the shadow curves in $\pi(S)$ are crossing, after shrinking, a shadow curve still meets itself or another shadow curve at a point q close to p. Thus, the vertical line ℓ through q intersects $\mathcal{B}'$ in more than one connected component. Note that q lies in an open region in $\pi(S)$ and therefore, ℓ intersects the interior of $\mathcal{B}$ in more than one connected component. ◻

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, of vertical edges of the polygon, and of parts of vertical facets. To correctly find the silhouette on the vertical facets, we use the vertical decomposition (or trapezoidal map) 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.

If we examine the possible intersections of two shadow edges, we find that there are four cases that have to be treated as crossings, as shown in Figure 8. Two shadow edges cross

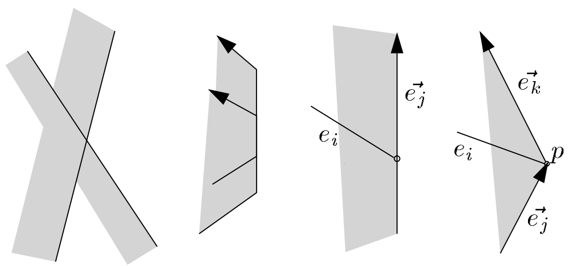

Figure 8:

if they intersect in their interior. They cross if they overlap with the same orientation (compare with Figure 4, where they overlap with opposite orientation—that is not a crossing). If an endpoint of edge e_i lies on e_j, and e_i lies to the left of e_j, this is a crossing. Finally, if an endpoint of e_i coincides with the destination of e_j, we have to test whether 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.

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 time $O(n \log n)$. Then, local analysis suffices to identify the silhouette curves. We project them into the plane to find the shadow curves. A plane sweep algorithm is used to compute the planar subdivision induced by the shadow curves. The algorithm finds all places where shadow edges meet. We test the edges for crossing. If no crossing exists, the subdivision has linear complexity and the algorithm terminates in time $O(n \log n)$. As soon as a crossing is found, we can also terminate, so the running time is $O(n \log n)$. ◻

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 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 set of 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 may cross the projection of an edge e—the set of these directions can be described by $O(n^2)$ great circle arcs. 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 $\mathcal{S}^2$. It is easily verified that inside every face of the arrangement, $\mathcal{P}$ is either z-monotone for every direction, 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 for 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 simply computing the intersections of all pairs of curves in $\mathcal{Q}$. This takes $O(n^4)$ time. Then for each computed vertex 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 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 searching for directions in the interior of cells of $\mathcal{A}(\mathcal{Q})$. We need however to consider all faces of the arrangement $\mathcal{A}(\mathcal{Q})$ and not only 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 cells in this arrangement is quite small, provided that the arrangement is in general position. In fact, 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 cells 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 total number of vertices in the arrangement. The algorithm is a straightforward adaptation of the plane-sweep paradigm to the sphere and its running time is $O((m + n^2) \log n)$. Its output is a data structure that allows for a traversal of the arrangement cell by adjacent cell (we could use, say, the quad-edge data structure for the purpose [10]).

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.

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 simple 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 cell of the arrangement. By the edges of the arrangement that are involved in the move, we know how to update the counters at a constant time per edge involved. At the end of the move we check the counters and report castability if they are both zero.

If the arrangement $\mathcal{A}(\mathcal{Q})$ is not in general position we may need to make several updates to the counters per move. This happens when several coplanar great circles or great circle arcs overlap. Before computing $\mathcal{A}(\mathcal{Q})$, the edges induced in the overlapping of coplanar great circles or great circle arcs are first computed in $O(n^2 \log n)$ time. We can also determine at the same time the updates to the counters when such an edge is crossed. Then we compute $\mathcal{A}(\mathcal{Q})$ as before. We conclude that all directions for which there is a good cast can be computed in $O(n^4)$ time, or in time $O((m + n^2) \log n)$.

There exist polyhedra that have as many as $\Omega(n^4)$ distinct cast directions. 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. An example of such a polyhedron can be found in the full paper.

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. Moreover, there are polyhedra for which there are $\Omega(n^4)$ distinct directions in which there is a good cast. Alternatively, all the directions in which there is a good cast can be computed in $O((m + n^2) \log n)$ time, where m is the number of vertices in the arrangement $\mathcal{A}(\mathcal{Q})$ defined above.*

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 9 shows a heart-shaped object. The sphere represents

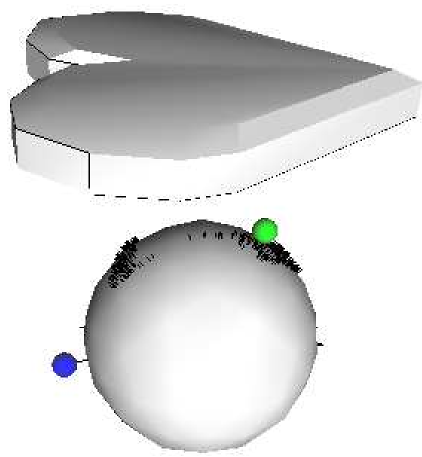

Figure 9:

the sphere of directions. A black stipple is plotted on the sphere for every direction in which the object has been found castable. Two directions have been specially marked, one of these is an edge direction, the other one a randomly chosen direction. The heart has 75 edges and 7 edge directions were found to be feasible. Of 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).

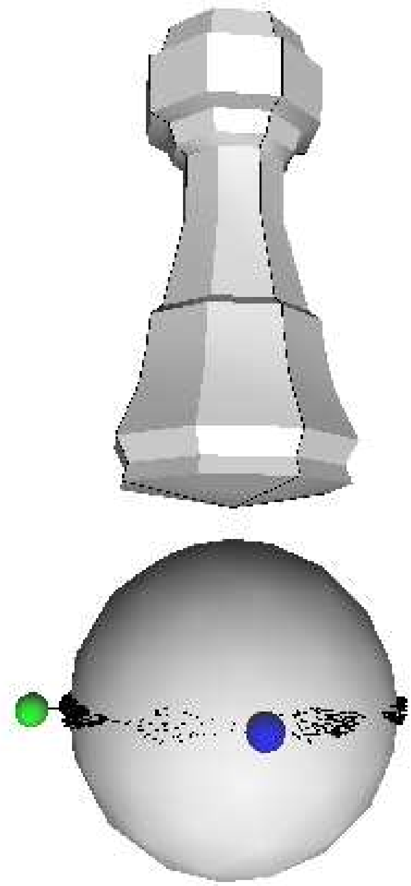

Figure 10:

The rook from a chess game in Figure 10 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, but so far it seems that that 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 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 moved in direction $\vec{d}_r$ to infinity without colliding with either $\mathcal{P}$ or $\mathcal{C}_b$, and $\mathcal{C}_b$ can be moved in direction $\vec{d}_b$ to infinity without colliding with $\mathcal{P}$. Contrary to the case of opposite removal directions, the order of removing the casts can be important. The cast induces partition of $\partial\mathcal{P}$ into two terrains, a red terrain (touching $\mathcal{C}_r$) and a blue terrain. When looking at $\mathcal{P}$ from infinity in direction $\vec{d}_r$ every point on the red terrain must be visible, and similar for the blue terrain. The reverse of this statement, however, is not true, as the next theorem shows.

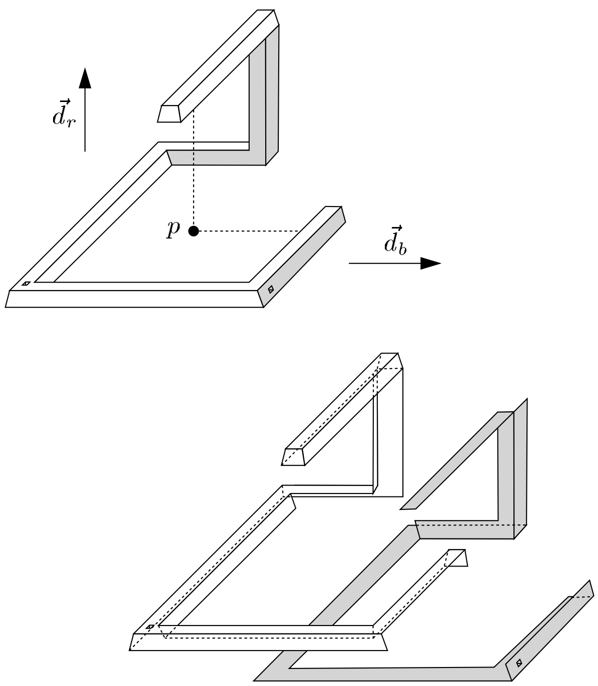

Figure 11: Illustration for the proof of Theorem 6.1

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, and which are completely visible from those directions.*

Proof: Consider the polyhedron $\mathcal{P}$ and the removal directions depicted in Figure 11. The shaded facets of $\mathcal{P}$ together with the bottom facets form the blue terrain on the boundary; the remaining facets form the red terrain. Both the blue and the red terrain are actually terrains in their respective removal directions. There is no good cast, however, for these removal directions for $\mathcal{P}$: The point p will collide with $\mathcal{P}$ when moved in direction $\vec{d}_b$ and when moved in $\vec{d}_r$, 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 11, we can ensure that there cannot be a good cast for any other pair of directions either. The hole in the blue terrain, for example, makes $\vec{d}_b$ the only direction for which this hole is part of a terrain. $\square$

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 be opposite, and gave necessary and sufficient conditions under which a cast exists. We also developed an algorithm to compute the cast for a polyhedral object.

We briefly touched on the case of removal directions that are not necessarily opposite, and showed that in that case it is not sufficient to argue about the surface of the polyhedron. Therefore, previous approaches do not generalize to that setting.

There are several interesting directions for further research. While we implemented a heuristically improved version of the algorithm that 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 objects whose boundary consists of, say, cubic B-spline patches. Finally, if an object is not castable in our setting, one would like to see whether it is castable if one is allowed to use so-called cores or inserts, or casts consisting of more than two parts. In practice, it is desirable to maximize the "flatness" of the parting surface between the two cast parts. Majhi et al. [15] 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.

References

[1] B. Asberg, G. Blanco, P. Bose, J. Garcia, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu. Feasibility of design in stereolithography. In *Proc. 13th Symp. on FST & TCS. Lecture Notes in Computer Science*, vol. 761, pages 228–237. Springer-Verlag, 1993. To appear in Algorithmica.

[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. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 123–131, 1994. To appear in Algorithmica.

[4] P. Bose, M. van Kreveld, and G. Toussaint. Filling polyhedral molds. In *Proc. 3rd Workshop Algorithms Data Struct. Lecture Notes in Computer Science*, vol. 709, pages 210–221. Springer-Verlag, 1993.

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

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

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

[8] S. Fekete and J. S. B. Mitchell. Geometric aspects of injection molding. Workshop on Geometric and Compu-

tational Aspects of Injection Molding, Bellairs Research Institute, February 5–12, 1993.

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

[10] L. 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.

[11] M. Held. *On the Computational Geometry of Pocket Machining. Lecture Notes in Computer Science*, vol. 500. Springer-Verlag, June 1991.

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

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

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

[15] 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.

[16] J. Matoušek. Efficient partiton trees. *Discrete & Computational Geometry*, 8:315–334, 1992.

[17] 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, Waterloo, Canada, 1993.

[18] M. Overmars, A. Rao, O. Schwarzkopf, and C. Wentink. Immobilizing polygons against a wall. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 29–38, 1995.

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

[20] 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.

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

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

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

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