

Geometric Algorithms for Layered Manufacturing

Ravi Janardan and Michiel Smid

ABSTRACT. Layered Manufacturing is a technology that allows physical prototypes of three-dimensional models to be built from their digital representation, as a stack of two-dimensional layers. One of the key problems is the choice of a suitable direction in which the model should be oriented and built, so as to minimize the number of layers, the stair-stepping effect, the volume of the support structures that are generated during the build, or the area of contact between the prototype and the support structures, or a combination of these measures. Other problems include orienting the model so that one or more prescribed facets are not in contact with supports, and determining a direction to fill in the individual layers. A final problem is to decompose the model into two submodels, which are built independently, so as to reduce the amount of support structures. In this survey, we give a detailed overview of efficient algorithms for compute problems. The algorithms use a large variety of techniques from computational geometry, such as convex hulls, Voronoi diagrams, spherical sweep, ray-shooting, Boolean operations on polygons, and arrangements.

1. Introduction

Layered Manufacturing is a fast-growing technology with significant impact on the efficiency of the design process in a broad range of industries; see Jacobs [Jac92], Kai and Fai [KF97], and Chua *et al.* [CLL03]. Layered Manufacturing offers a flexible and cost-effective alternative to traditional methods used in the design phase of physical prototypes. Currently, this technology produces high-quality prototypes in a matter of hours and at low cost. The prototypes can be inspected for flaws and if necessary the design can be modified and the process repeated until the final design has reached the desired quality.

Stereolithography is a widely-used Layered Manufacturing process. The Stereolithography Apparatus consists of a vat of light-sensitive liquid resin, a platform, and a laser; see Figure 1. The input to the process is a surface triangulation of the digital model in the industry-standard STL-format. This format consists of an unordered list of triangles, each specified by the coordinates of its vertices, and the unit outer-normal of each triangle. The model is sliced into horizontal two-dimensional layers, which are then sent over a network to the Stereolithography Apparatus. The laser traces out the contour of each layer (which is a polygon)

1991 *Mathematics Subject Classification.* Primary 68U05; Secondary 68U07.

Key words and phrases. Computational geometry, computer aided design.

The second author was supported by NSERC.

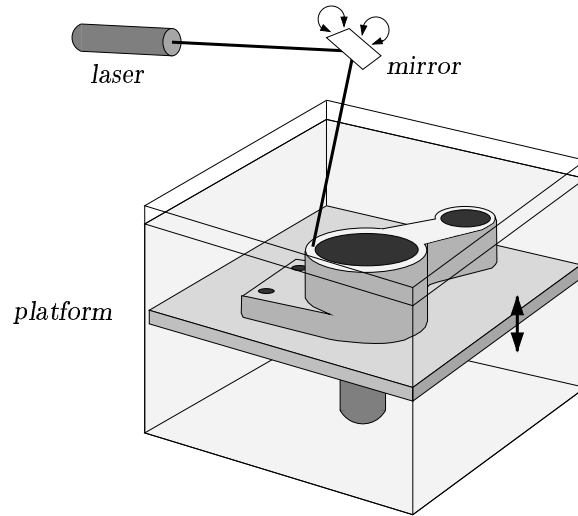


FIGURE 1. *The Stereolithography Apparatus.*

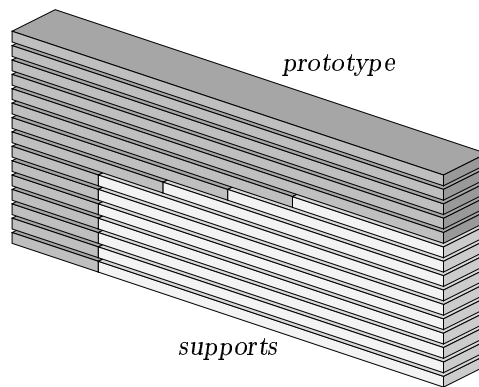


FIGURE 2. *Illustrating support structures. The model is in dark gray, whereas the supports are in light-gray.*

and then scans the interior in a zig-zag pattern. The exposure to the laser causes the scanned portion of the liquid to harden and form the physical layer. The platform is then lowered by an amount equal to the layer thickness (typically a few thousandths of an inch) and the next layer is then built on top of the previous one. Thus, the three-dimensional prototype is realized eventually as a vertical stack of two-dimensional layers. Ideally each new layer should rest completely on top of the previous one, so that the prototype is self-supporting during the build phase. The complex shape of real-world prototypes, however, often prevents them from being self-supporting. Therefore, during a pre-processing step, the model is analyzed and additional structures called *supports* are created and merged with the description of the model; see Figure 2. These support structures are built simultaneously with the prototype and removed later.

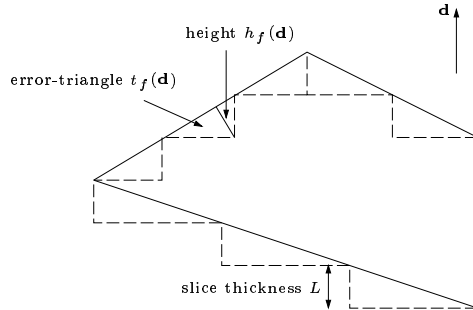


FIGURE 3. *Illustrating stair-stepping. For ease of depiction, the underlying model is assumed to be a prism of uniform cross-section perpendicular to the plane of the paper. The polygon shown is the cross-section of the prism.*

The Layered Manufacturing consists of three phases: preprocessing, building, and postprocessing. Preprocessing includes repairing flaws in the STL file (e.g., gaps between facets, geometric singularities, etc.), deciding upon a suitable initial orientation for the model, computing support requirements, generating and merging a description of the supports into the STL file, and slicing the model and supports. The building phase involves tracing and filling in each layer. Postprocessing includes removal of supports and improving part finish and accuracy.

1.1. Geometric issues. A key step in Layered Manufacturing is choosing an orientation for the model, i.e., the *build direction*. Its choice can impact critically the efficiency and cost of the build process as well as the surface quality of the physical prototype. Below, we mention several competing criteria that need to be addressed when choosing a build direction.

1.1.1. *Number of layers.* In the Layered Manufacturing process, the layer thickness is typically measured in thousandths of an inch. Because of this, the number of layers needed to build a model can run into the thousands if the part is oriented along its longest dimension. If we assume the layer thickness to be fixed, then the number of layers for a given build direction \mathbf{d} is proportional to the smallest distance between two parallel planes that are normal to \mathbf{d} and that enclose the model. Hence, orienting the model so as to minimize the number of layers is equivalent to determining a direction \mathbf{d} for which the distance between the enclosing planes is minimum; this minimum distance is called the *width*.

1.1.2. *Stair-stepping effect.* Due to the non-zero layer thickness, the physical prototype will have a stair-stepped finish on each facet f that is not parallel to the build direction \mathbf{d} . The degree of stair-stepping on f depends on the angle between the normal of f and \mathbf{d} , and it can be mitigated by a suitable choice of \mathbf{d} . Bablani and Bagchi [BB95] introduce the notion of an *error-triangle* as a way of quantifying the amount of stair-stepping; see Figure 3. They argue that a good build direction is one for which the maximum height of any error-triangle (over all facets of the model) is minimized.

1.1.3. *Support structures.* If \mathbf{d} is a direction and f is a facet of \mathcal{P} , then we say that f is a *back facet* with respect to \mathbf{d} , if the angle between the outer unit-normal \mathbf{n}_f of f and \mathbf{d} is greater than $\pi/2$. If this angle is less than $\pi/2$, then we say that

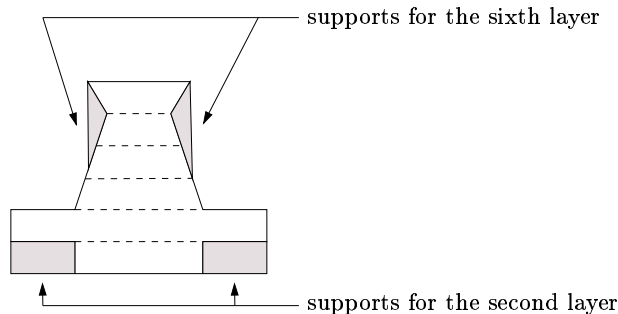


FIGURE 4. Illustrating support structures, shown in 2D for convenience.

f is a *front facet* with respect to \mathbf{d} . Finally, if this angle is equal to $\pi/2$, then we say that f is a *parallel facet* with respect to \mathbf{d} .

All back facets of \mathcal{P} will need to be supported. Consider such a back facet f with respect to a build direction \mathbf{d} . The *support polyhedron* for f is the closure of the set of all points $p \in \mathbb{R}^3$ such that p is not in the interior of \mathcal{P} and the ray shot from p in direction \mathbf{d} first enters \mathcal{P} through f . Informally, the support polyhedron of f is bounded from above by f , on the sides by vertical facets that “drop down” from the edges of f , and from below by the platform on which \mathcal{P} rests and/or portions of front facets of \mathcal{P} . (Observe that if \mathcal{P} is convex, then it is bounded from below by only the platform.) The *supports* of \mathcal{P} with respect to a build direction is the collection of support polyhedra for all back facets; see Figure 4.

The *contact-area* is the total surface area of \mathcal{P} that is in contact with supports. It includes the area of all the back facets of \mathcal{P} and the areas of those portions of front facets and parallel facets that are in contact with supports.

Observe that for a convex polyhedron, the support structures are relatively simple, in that only back facets are in contact with supports and every point on a back facet is in contact with supports. Furthermore, the support structures extend all the way down to the platform. For a general polyhedron, however, the situation is more complex: In addition to back facets, some front and parallel facets can also be in contact with supports, and the latter two types of facets may be only partially in contact. In addition, supports need not extend all the way down to the platform, but may instead terminate on other parts of the polyhedron. This is illustrated in Figure 4 for the supports for the sixth layer.

A build direction that minimizes the total volume of all supports will translate into a faster build time and less waste of material. Similarly, a build direction that minimizes the contact-area will minimize the portions of the prototype that are affected during the post-processing stage and help minimize damage to the surface of the prototype during support removal.

1.1.4. *Protecting a facet from being in contact with supports.* The removal of supports from a facet affects the surface quality and accuracy adversely, thereby impacting the functional properties of critical facets, such as, for instance, facets on gear teeth. Therefore, a good build direction is one for which one or more prescribed facets are not in contact with supports.

1.1.5. *Decomposing the model.* Traditionally, a process-planning algorithm for Layered Manufacturing works by viewing the model as a single, monolithic unit.

As a result, large models that cannot be accommodated in the workspace cannot be constructed. A possible approach is to decompose the model into a small number of pieces, build each piece separately, and then glue the built pieces together to obtain the physical prototype. Besides the advantage that larger models can be built, the amount of time needed to build a model is reduced in this way. Another crucial advantage is that the support requirements of this decomposition-based approach are often much less than those of the conventional approach. For instance, if a hollow sphere is built the conventional way, supports will be needed in the interior void and below the lower hemisphere; however, if it is built as two hemispheric shells, in opposite directions, then supports will be needed only in the regions previously occupied by the void, which results in an overall reduction in both support volume and contact-area. This leads to the problem of how to decompose a model and build it so that the overall support requirements are minimized.

1.1.6. *Hatching.* The process of printing the layers is called *hatching*. This is essentially a two-dimensional problem, since each layer is a polygon (possibly with holes). The hatching process depends on the specific Layered Manufacturing process being used. In Stereolithography, the laser traces out the boundary of the polygon and then fills in the interior by following paths along equally-spaced parallel lines, switching itself on (resp. off) when it is in the interior (resp. reaches the boundary) of the polygon. The result is a set of parallel line segments, called *hatching segments*, in the interior of the polygon, where each segment has a fixed width determined by the width of the tool-tip, e.g., the laser. (This width is very small compared to the segment length.) The speed of the hatching process depends heavily on the number of hatching segments. This number is, in turn, a function of the direction (in the plane) followed by the laser; this direction is called the *hatching direction*. This leads to the problem of computing a hatching direction that minimizes the number of hatching segments.

1.2. Geometric optimization problems. The geometric issues introduced above lead to the problem of computing a build direction (or hatching direction) that optimizes one or more of the criteria. In this paper, we show that, for each of the criteria, the optimization problem can be formulated in purely geometric terms. We then show how techniques from computational geometry can be used to solve these problems efficiently. In particular, we use techniques such as convex hulls, Voronoi diagrams, spherical sweep, ray-shooting, Boolean operations on polygons, and arrangements.

1.3. Notation. Throughout this paper, we denote by \mathcal{P} the polyhedral model of interest, and by n the number of its facets. We assume that the facets of \mathcal{P} are triangles and that its boundary is represented in some standard form, such as, for instance, a doubly-connected edge list (see de Berg *et al.* [dBvKOS00]) or a winged-edge structure (see Baumgart [Bau75]). If necessary, such a representation can be computed easily from the standard STL representation of \mathcal{P} ; see McMains and Séquin [MS99].

In the problems considered in this paper, we want to compute a build direction \mathbf{d} that optimizes one or more criteria. We will represent directions as points—or unit-vectors—on the *unit-sphere* \mathbb{S}^2 , i.e., the boundary of the three-dimensional ball centered at the origin and having radius one. The *upper hemisphere* is defined as

$$\mathbb{S}_+^2 := \mathbb{S}^2 \cap \{(x, y, z) \in \mathbb{R}^3 : z \geq 0\}.$$

Similarly, we define the *lower hemisphere* as

$$\mathbb{S}_-^2 := \mathbb{S}^2 \cap \{(x, y, z) \in \mathbb{R}^3 : z \leq 0\}.$$

Finally, the *equator* is the intersection of \mathbb{S}^2 with the plane $z = 0$.

2. Minimizing the number of layers

Assuming that the layer thickness is fixed, the number of layers for a given build direction \mathbf{d} is proportional to the smallest distance between two parallel planes that are normal to \mathbf{d} and that enclose the model \mathcal{P} . We call this smallest distance the *width of \mathcal{P} in direction \mathbf{d}* , and denote it by $W(\mathbf{d})$. Observe that $W(\mathbf{d}) = W(-\mathbf{d})$. The *width $W(\mathcal{P})$* of the polyhedron \mathcal{P} is defined as the minimum distance between any two parallel planes that enclose \mathcal{P} , i.e.,

$$W(\mathcal{P}) = \min\{W(\mathbf{d}) : \mathbf{d} \in \mathbb{S}^2\}.$$

We consider the problem of computing a build direction \mathbf{d} for which $W(\mathbf{d}) = W(\mathcal{P})$. The problem of computing the width of \mathcal{P} was considered by Houle and Toussaint [HT88]. We outline their approach.

First observe that the width of the polyhedron \mathcal{P} is equal to the width of the convex hull $CH(\mathcal{P})$ of \mathcal{P} . Therefore, it suffices to consider the convex polyhedron $CH(\mathcal{P})$.

Let v be any vertex and let f be any facet of $CH(\mathcal{P})$. We say that (v, f) is an *antipodal vertex-facet pair* (or *VF-pair*), if the two parallel planes containing v and f , respectively, enclose $CH(\mathcal{P})$. Similarly, two *non-parallel* edges e_0 and e_1 of $CH(\mathcal{P})$ are said to be an *antipodal edge-edge pair* (or *EE-pair*), if the two parallel planes containing e_0 and e_1 , respectively, enclose $CH(\mathcal{P})$.

Houle and Toussaint prove that any direction \mathbf{d} that is not associated with some *VF-* or *EE-*pair can be rotated to a direction \mathbf{d}' such that $W(\mathbf{d}') < W(\mathbf{d})$. In other words, any direction minimizing the width of \mathcal{P} is perpendicular to the parallel planes associated with some *VF-* or *EE-*pair. Therefore, the width of \mathcal{P} can be obtained by first computing all *VF-* and *EE-*pairs, and then for each of them computing the distance between the corresponding supporting parallel planes. The smallest distance found is equal to the width $W(\mathcal{P})$ of the polyhedron \mathcal{P} .

In order to compute all *VF-* and *EE-*pairs, we consider the *dual graph G* of $CH(\mathcal{P})$, which is the planar graph on the unit-sphere \mathbb{S}^2 that is defined as follows. The vertices of G are the outer unit-normals of the facets of $CH(\mathcal{P})$. Two of these vertices define an edge in G if the corresponding facets of $CH(\mathcal{P})$ share an edge. Edges of this dual graph are drawn as great arcs on \mathbb{S}^2 . Observe that edges (resp. faces) of G are in one-to-one correspondence with edges (resp. vertices) of $CH(\mathcal{P})$. It turns out to be convenient to transform the graph G into a planar graph G' on \mathbb{S}^2 , by cutting all edges that cross the equator, and “adding” the equator to it.

Let G'_u be the subgraph of G' consisting of all vertices and edges that are in the upper hemisphere \mathbb{S}_+^2 . Let G'_ℓ be the subgraph of G' consisting of all vertices and edges that are in the lower hemisphere \mathbb{S}_-^2 . Finally, let G'_ℓ be the *inverse image* of G'_ℓ , i.e., the graph obtained by mapping each vertex \mathbf{v} of G'_ℓ to the vertex $-\mathbf{v}$. Both graphs G'_u and G'_ℓ are in the upper hemisphere \mathbb{S}_+^2 .

We now show how the graphs G'_u and G'_ℓ can be used to obtain all *VF-* and *EE-*pairs.

2.1. Computing VF-pairs. Consider an arbitrary VF-pair (v, f) . Let f_v be the face of G that corresponds to v , and let \mathbf{d}_f be the vertex of G that corresponds to f . There are two cases to consider. The first case is when \mathbf{d}_f is on or above the equator. Then \mathbf{d}_f is a vertex of G'_u . Let f_v^0 be the face of G'_ℓ that is contained in f_v , and let f'_v be the face of G'_ℓ that corresponds to f_v^0 . Since the unique planes that support $CH(\mathcal{P})$ at v and f are parallel, vertex \mathbf{d}_f of G'_u is contained in face f'_v of G'_ℓ . The second case is when \mathbf{d}_f is below the equator. In this case, \mathbf{d}_f is a vertex of G'_ℓ , and $-\mathbf{d}_f$ is a vertex of G'_u . Let f'_v be the face of G'_u that is contained in f_v . Then vertex $-\mathbf{d}_f$ of G'_u is contained in face f'_v of G'_u . We have proved the following result.

LEMMA 2.1. *We can obtain all VF-pairs by performing a point location query with each vertex of G'_u in the graph G'_ℓ , and by performing a point location query with each vertex of G'_ℓ in the graph G'_u . In both these point location problems, all query points are known in advance.*

2.2. Computing EE-pairs. Consider two arbitrary edges e_0 and e_1 of $CH(\mathcal{P})$ that form an EE-pair. Let g_0 and g_1 be the edges of G that correspond to e_0 and e_1 , respectively. We may assume without loss of generality that g_0 is (completely or partially) contained in the upper hemisphere. Hence, g_1 is (again, completely or partially) contained in the lower hemisphere. Let g'_0 be the part of g_0 that is contained in \mathbb{S}^2_+ . Then g'_0 is an edge of G'_u . Let g'_1 be the part of g_1 that is contained in \mathbb{S}^2_- , and let g_1 be its inverse image. Then g_1 is an edge of G'_ℓ . Since the unique planes that support $CH(\mathcal{P})$ at e_0 and e_1 are parallel, the edges g'_0 and g_1 intersect. Hence, we have proved the following result.

LEMMA 2.2. *We can obtain all EE-pairs by computing all intersections between edges of G'_u and edges of G'_ℓ .*

2.3. The algorithm. Lemmas 2.1 and 2.2 imply that all VF- and EE-pairs can be obtained from the overlay of the graphs G'_u and G'_ℓ . Since this overlay is defined by $O(n)$ great arcs, it can be computed in $O(n^2)$ time. For details, see Houle and Toussaint [HT88]. Below, we give an alternative algorithm and express its running time in terms of (i) the number n of facets of \mathcal{P} , (ii) the number h of facets of $CH(\mathcal{P})$, and (iii) the number k of intersections between edges of G'_u and edges of G'_ℓ . Observe that, in the worst case, $h = \Theta(n)$ and $k = \Theta(n^2)$.

The convex hull of the polyhedron \mathcal{P} can be computed in $O(n \log n)$ time, see, e.g., de Berg *et al.* [dBvKOS00]. Observe that the graphs G'_u and G'_ℓ have total size $O(h)$. Based on Lemmas 2.1 and 2.2, we obtain all VF- and EE-pairs by (i) computing for each vertex of one of these graphs, the face of the other graph that contains the vertex, and (ii) computing all intersections between edges of these graphs. Both these problems can be solved in $O(h \log h + k \log h)$ time, using the Bentley–Ottmann sweep algorithm [BO79], adapted to great arcs on the unit-sphere. The overall running time of the algorithm is thus $O(n \log n + k \log h)$, which is $O(n^2 \log n)$ in the worst case.

THEOREM 2.3. *Let \mathcal{P} be a polyhedron with n facets. A build direction that minimizes the number of layers can be computed in $O(\min\{n^2, n \log n + k \log h\})$ time, where h is the number of facets of the convex hull of \mathcal{P} and k is the number of intersections between edges of the graph G'_u and edges of the graph G'_ℓ . In the worst case, $h = \Theta(n)$ and $k = \Theta(n^2)$.*

An implementation of the algorithm described above, together with extensive experimental results, can be found in Schwerdt *et al.* [SSMJ99]. These experiments show that the value of k is usually less than h . That is, the running time on practical instances is much less than the theoretical near-quadratic worst-case upper bound.

Gärtner and Herrmann [GH01] present an implementation of an alternative algorithm that finds all VF - and EE -pairs without using the Bentley–Ottmann sweep technique. They reformulate the width problem as an optimization problem with linear constraints and a non-convex objective function. This new algorithm has a worst-case running time of $\Theta(n^2)$. Experimental results show that it usually outperforms the implementation of [SSMJ99].

We finally remark that the asymptotically fastest known algorithm for computing the width of a three-dimensional point set is due to Agarwal and Sharir [AS96]; its expected running time is close to $O(n^{1.5})$. The disadvantage of their algorithm is that it computes only one direction that minimizes the width and that it is probably extremely difficult to implement. In contrast, the algorithm described in this section finds *all* such directions. (Houle and Toussaint [HT88] have shown that there can be $\Theta(n^2)$ many such directions.)

3. Minimizing the stair-stepping effect

In this section, we consider the problem of computing a build direction that minimizes the maximum height of any error-triangle.

Let L denote the layer thickness and, for each facet f of the polyhedron \mathcal{P} , let $h_f(\mathbf{d})$ denote the height of the error-triangle $t_f(\mathbf{d})$ for facet f ; see Figure 3. Our problem is that of computing a build direction \mathbf{d} that minimizes the maximum error-triangle height $ETH(\mathbf{d}) := \max_f h_f(\mathbf{d})$. We will transform this min-max problem into a max-min problem.

Let $\theta'_f(\mathbf{d})$ be the angle between \mathbf{d} and the outer unit-normal \mathbf{n}_f of f , and let $\theta''_f(\mathbf{d})$ be the angle between \mathbf{d} and $-\mathbf{n}_f$. We define $\theta_f(\mathbf{d}) := \min\{\theta'_f(\mathbf{d}), \theta''_f(\mathbf{d})\}$. Since $h_f(\mathbf{d}) = L \cos \theta_f(\mathbf{d})$, the problem of minimizing $ETH(\mathbf{d})$ is equivalent to computing a build direction \mathbf{d} that maximizes the value of $\min_f \theta_f(\mathbf{d})$. Let us see how the latter problem can be solved. Consider the set

$$S := \{\mathbf{n}_f : f \text{ is a facet of } \mathcal{P}\} \cup \{-\mathbf{n}_f : f \text{ is a facet of } \mathcal{P}\}.$$

Maximizing the value of $\min_f \theta_f(\mathbf{d})$ means that we want to compute a direction \mathbf{d} for which the minimum angle between \mathbf{d} and the elements of S is maximized.

A *cap* on \mathbb{S}^2 , with *pole* \mathbf{d} and *radius* θ , is the set of all points on \mathbb{S}^2 whose distances from \mathbf{d} are less than or equal to θ , as measured along the surface of \mathbb{S}^2 . Hence, our problem is equivalent to computing a cap whose interior does not contain any element of S and whose radius is maximum, i.e., a *largest empty cap*. The pole of this cap is the desired optimal direction.

Since a largest empty cap must have at least three elements of S on its boundary, it is easy to compute such a cap in $O(n^3)$ time. However, it is possible to derive a significantly faster algorithm, with a running time of $O(n \log n)$, based on the following properties.

1. Let c be the circle bounding a cap C and let $H(C)$ be the plane such that $c = H(C) \cap \mathbb{S}^2$. If C is empty, then all elements of S lie on the same side of $H(C)$. Conversely, every plane H that intersects \mathbb{S}^2 and for which all elements of S lie on the same side of H corresponds to an empty cap.

2. The larger the cap C is, the closer is $H(C)$ to the origin.
3. A largest empty cap C must have at least three elements of S on its boundary, i.e., $H(C)$ contains a facet of the convex hull of S .

The discussion above suggests the following algorithm. First compute the set S and its convex hull $CH(S)$. For each facet g of $CH(S)$, determine the plane H_g through g . Among all these planes H_g , compute one that is closest to the origin. Finally, compute the normal \mathbf{d} from the origin to this closest plane. This direction \mathbf{d} has the property that when \mathcal{P} is built in direction \mathbf{d} , the maximum error-triangle height $ETH(\mathbf{d})$ is minimized. Observe that the running time of this algorithm is dominated by the $O(n \log n)$ time needed for the convex hull computation; see de Berg *et al.* [dBvKOS00].

THEOREM 3.1. *Let \mathcal{P} be a polyhedron with n facets. A build direction that minimizes the maximum error-triangle height can be computed in $O(n \log n)$ time.*

As we saw, the correctness of the algorithm follows from the fact that a direction \mathbf{d} minimizing $ETH(\mathbf{d})$ must be the normal vector of some facet of the convex hull of S . This is equivalent to saying that \mathbf{d} is a vertex of the *spherical Voronoi diagram* of S . (In this Voronoi diagram, distances are measured along \mathbb{S}^2 .)

The algorithm presented in this section is due to Majhi *et al.* [MJSG99]. These authors show how the algorithm can be extended to the case when each facet of \mathcal{P} is assigned a positive weight, to reflect its importance in the model, and the objective is to minimize the maximum weighted error-triangle height. This version of the problem can also be solved in $O(n \log n)$ time. They also show how the sum of the weighted error-triangle heights, taken over all facets, can be minimized in $O(n^2)$ time.

4. Optimization of support structures

Asberg *et al.* [ABB⁺97] give an algorithm that decides, in $O(n)$ time, if there exists a build direction \mathbf{d} for a given polyhedron with n facets, such that no support structures are needed. In this section, we consider the more general problem of minimizing the support structures. For simplicity, we restrict ourselves to the case when we want to minimize the contact-area of the supports. The approach presented here can be generalized to the case when the goal is to minimize the volume of the support structures.

Let \mathcal{P} be a polyhedron with n facets and let $\mathbf{d} \in \mathbb{S}^2$ be a direction. Recall that the *contact-area for build direction \mathbf{d}* is defined as the total area of all points on the boundary of \mathcal{P} that are in contact with supports, when \mathcal{P} is built in direction \mathbf{d} . The problem we consider is that of computing a build direction \mathbf{d} for which the contact-area is minimum.

4.1. The convex case. We first show how the case when \mathcal{P} is a convex polyhedron can be solved. The algorithm is due to Majhi *et al.* [MJSG99]. Recall the definition of back facets and front facets, see Section 1.1.3. Observe that the contact-area for build direction \mathbf{d} is equal to the total area of all back facets.

The set of all directions \mathbf{d} for which a facet f is a back facet can be represented by an open hemisphere C_f on \mathbb{S}^2 , whose pole is the point $-\mathbf{n}_f$. We associate with this hemisphere a weight that is equal to the area of f . Our problem is now equivalent to computing a point on \mathbb{S}^2 such that the total weight of all hemispheres C_f containing this point is minimized. This problem can be solved by computing and

traversing the arrangement defined by the boundaries of all these hemispheres; the time and space requirements will both be $O(n^2)$. Using the topological sweep algorithm of Edelsbrunner and Guibas [EG89], the space requirement can be reduced to $O(n)$.

THEOREM 4.1. *Let \mathcal{P} be a convex polyhedron with n facets. A build direction that minimizes the total surface area of \mathcal{P} that is in contact with supports can be computed in $O(n^2)$ time using $O(n)$ space.*

The time bound in Theorem 4.1 is probably optimal, because, as shown in Majhi *et al.* [MJSG97], the following closely related problem belongs to the class of *3SUM-hard problems* (see Gajentaan and Overmars [GO95]): Given a convex polyhedron, compute a direction with a positive z -component that minimizes the number of back facets.

In related work, Agarwal and Desikan [AD00] consider the problem of approximating the minimum contact-area. They show that, for a convex polyhedron, the minimum contact-area can be approximated to within a factor of $1 + \epsilon$, in $O((n/\epsilon^3) \log^3 n)$ expected time.

When a three-dimensional model is built using Layered Manufacturing, usually an entire facet is in contact with the platform, because it provides more stability to the part. Majhi *et al.* [MJSG99] show how to transform the contact-area problem for this special case to a halfplane range counting problem on weighted points in two-dimensional space. Using a result of Matoušek [Mat93], they show that, for a convex polyhedron \mathcal{P} , a build direction for which a facet rests on the platform and that minimizes the total surface area in contact with supports, can be computed in close to $O(n^{4/3})$ time. Agarwal and Desikan [AD00] show that the approximation version of this problem can be solved in $O((n/\epsilon^2) \log^2 n)$ expected time.

As a final remark, Agarwal and Desikan [AD00] show the following: Given a (possibly non-convex) polyhedron \mathcal{P} and a build direction \mathbf{d} , the total area of all facets that are in contact with supports can be computed in close to $O(n^{4/3})$ time. Observe, however, that this is not the same as the total contact-area.

4.2. The general case. We now consider the contact-area minimization problem for a general (i.e., not necessarily convex) polyhedron \mathcal{P} . We remark that this problem is highly non-trivial, even in the two-dimensional case (i.e., when \mathcal{P} is a simple polygon and the two-dimensional equivalent of supports is considered); an (inefficient) algorithm for this case can be found in Majhi *et al.* [MJS⁺99]. In fact, Agarwal and Desikan [AD00] argue that it is unlikely that the three-dimensional version can be solved in $o(n^4)$ time. In this section, we outline an algorithm due to Schwerdt [Sch01].

The basic approach for solving the problem is by subdividing the unit-sphere \mathbb{S}^2 into regions such that for each region I , the combinatorial structure of the supports is the same for all directions $\mathbf{d} \in I$. (This is related to the notion of the *aspect graph* of \mathcal{P} , see Bowyer and Dyer [BD90] and Plantinga and Dyer [PD90].) Given this subdivision of \mathbb{S}^2 , we derive an expression (as a function of \mathbf{d}) for the contact-area for each region I . Finally, for each region I , we minimize the expression over all directions $\mathbf{d} \in I$. The minimum value over all regions gives the minimum contact-area over all $\mathbf{d} \in \mathbb{S}^2$.

Observe that, for any given build direction \mathbf{d} , a back facet of \mathcal{P} is completely in contact with supports. On the other hand, a front facet is either not in contact

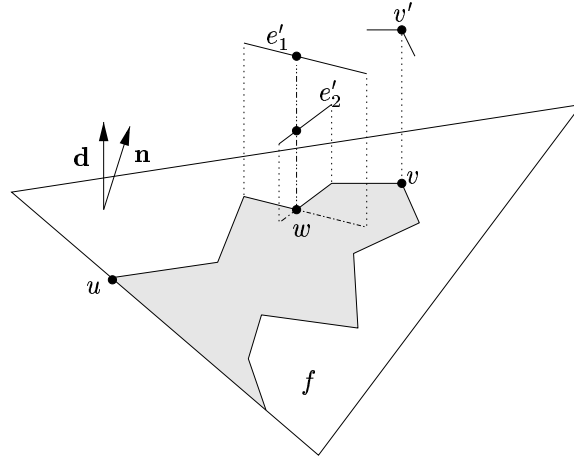


FIGURE 5. A support polygon on facet f . \mathbf{n} is the outer normal of f . u , v and w illustrate the three different types of vertices of a support polygon. e'_1 and e'_2 are edges and v' is a vertex of the polyhedron \mathcal{P} .

with supports, or is completely or partially in contact. It turns out that the partial contact case is the most difficult to handle.

4.2.1. *Subdividing \mathbb{S}^2* . Let f be a front facet that is partially in contact with supports. The boundary of the region on f that is in contact with supports consists of one or more polygons. We call each such polygon a *support polygon*; see Figure 5. Each vertex of a support polygon is either (i) the projection in direction $-\mathbf{d}$ onto f of a vertex of \mathcal{P} (see vertex v in Figure 5), or (ii) the intersection of the projections in direction $-\mathbf{d}$ onto f of two edges of \mathcal{P} (see vertex w in Figure 5). Vertex u in Figure 5 corresponds to a special case of (ii): For this vertex, one of the edges is not a projection, but an edge of f .

As mentioned already, our approach is to subdivide \mathbb{S}^2 into regions such that within each region, the expression for the total contact-area does not “change”. Directions for which the expression does change will be called *critical directions*; these will form the boundaries of the regions.

Assume we move the build direction \mathbf{d} along \mathbb{S}^2 . Then \mathbf{d} becomes a critical direction if a front facet becomes a back facet (or vice versa) or the combinatorial structure of some support polygon changes. A facet f changes from front facet to back facet (or vice versa) if \mathbf{d} crosses the great circle on \mathbb{S}^2 consisting of all directions that are parallel to f . The combinatorial structure of the support polygons on facet f can change in any one of the following situations:

1. \mathbf{d} is parallel to f .
2. A new support polygon appears.
3. An existing support polygon disappears.
4. Two support polygons combine into one support polygon.
5. A support polygon splits into two support polygons.
6. A new vertex appears on a support polygon.
7. A vertex on a support polygon disappears.

We now derive necessary conditions for each of these seven cases to occur. If \mathbf{d} is parallel to facet f (case 1 above), then a new support polygon may appear on f or an existing support polygon on f may disappear. Therefore, we consider the plane through f and translate it to the origin. The intersection of this translated plane with \mathbb{S}^2 gives all directions that are parallel to f . This covers case 1. Cases 2–7 are covered by Cases A and B below.

Case A: The projection in direction $-\mathbf{d}$ onto f of a vertex of \mathcal{P} is on the projection in direction $-\mathbf{d}$ onto f of an edge of \mathcal{P} .

In order to obtain all directions in this case, we do the following. For each vertex v and each edge e of \mathcal{P} , we consider the wedge consisting of all rays emanating from v that contain some point of e . The directions of all these rays are the critical directions for Case A. In this way, we obtain for each vertex-edge pair of \mathcal{P} a great arc on \mathbb{S}^2 .

Case B: The projections in direction $-\mathbf{d}$ onto f of three edges of \mathcal{P} intersect in one single point.

The critical directions for this case are obtained as follows. We want those directions \mathbf{d} such that a line with direction \mathbf{d} exists that intersects three edges of \mathcal{P} . Let (a_1, b_1) , (a_2, b_2) , and (a_3, b_3) be three edges of \mathcal{P} , and let p be a point on one of these edges. Let us assume that p is a point on (a_1, b_1) . Then $\mathbf{p} = \mathbf{a}_1 + \lambda(\mathbf{b}_1 - \mathbf{a}_1)$, for some λ with $0 \leq \lambda \leq 1$. Let E' be the plane through p and (a_2, b_2) , and let E'' be the plane through p and (a_3, b_3) . Then \mathbf{d} is a critical direction if E' and E'' intersect in a line having direction \mathbf{d} . The planes E' and E'' have normal vectors $(\mathbf{p} - \mathbf{a}_2) \times (\mathbf{b}_2 - \mathbf{a}_2)$ and $(\mathbf{p} - \mathbf{a}_3) \times (\mathbf{b}_3 - \mathbf{a}_3)$, respectively. The direction \mathbf{d} of the intersection of these planes is given by

$$\mathbf{d} = [(\mathbf{p} - \mathbf{a}_2) \times (\mathbf{b}_2 - \mathbf{a}_2)] \times [(\mathbf{p} - \mathbf{a}_3) \times (\mathbf{b}_3 - \mathbf{a}_3)].$$

Since each coordinate of the direction \mathbf{d} is a quadratic function in λ , \mathbf{d} is contained in a curve on \mathbb{S}^2 . (This curve will not be a great arc.)

Over all facets f of \mathcal{P} , case 1 gives rise to n great circles on \mathbb{S}^2 , whereas Case A gives $O(n^2)$ great arcs. For the critical directions in Case B, we obtain $O(n^3)$ curves on \mathbb{S}^2 , one curve for each triple of edges of \mathcal{P} . It can be shown that any two of these curves intersect only $O(1)$ times, and any such curve has only $O(1)$ intersections with any great circle.

The arrangement on \mathbb{S}^2 of all these n great circles, $O(n^2)$ great arcs, and $O(n^3)$ curves, subdivides \mathbb{S}^2 into $O(n^6)$ regions. The total number of vertices and edges of all these regions is $O(n^6)$. Moreover, the boundary of each such region consists of $O(n^3)$ edges.

4.2.2. The expression for the contact-area. We start by deriving expressions for the vertices of the support polygons. Let I be one of the regions in the arrangement of Section 4.2.1, and let f be a facet of \mathcal{P} that is a front facet for all directions in I . Let \mathbf{n} be the outer unit-normal of f , and let S be a support polygon on f . Recall that for all $\mathbf{d} \in I$, the polygon S has the same combinatorial structure. Consider an arbitrary vertex v of S . As we saw before, there are two cases to distinguish.

Case 1: v is the projection in direction $-\mathbf{d}$ onto f of a vertex v' of \mathcal{P} .

We can write $v = v' - \lambda\mathbf{d}$, for some $\lambda > 0$. Since the plane through f has the form $\mathbf{n} \cdot \mathbf{x} = c$, for some constant c , we obtain $\mathbf{n} \cdot (v' - \lambda\mathbf{d}) = c$, which can be

rewritten as

$$\lambda = \frac{\mathbf{n} \cdot v' - c}{\mathbf{n} \cdot \mathbf{d}}.$$

Hence, we have written v as

$$(4.1) \quad v = v' - \frac{\mathbf{n} \cdot v' - c}{\mathbf{n} \cdot \mathbf{d}} \mathbf{d}.$$

Case 2: v is the intersection of the projections in direction $-\mathbf{d}$ onto f of two edges of \mathcal{P} .

Let $e'_1 = (u'_1, v'_1)$ and $e'_2 = (u'_2, v'_2)$ be two edges of \mathcal{P} whose projections e_1 and e_2 , respectively, onto f in direction $-\mathbf{d}$ intersect in v . Let E_1 be the plane through e'_1 and e_1 , and let E_2 be the plane through e'_2 and e_2 . Observe that E_1 contains the points u'_1, v'_1 and $u'_1 - \mathbf{d}$. Moreover, E_1 has normal vector $\mathbf{d} \times (v'_1 - u'_1)$. Therefore, E_1 is given by the equation

$$[\mathbf{d} \times (v'_1 - u'_1)] \cdot \mathbf{x} = c_1,$$

for some constant c_1 . Since u'_1 is contained in E_1 , we have

$$c_1 = [\mathbf{d} \times (v'_1 - u'_1)] \cdot u'_1 = [\mathbf{d} \times v'_1] \cdot u'_1.$$

Hence, we have obtained the following equation for the plane E_1 :

$$[\mathbf{d} \times (v'_1 - u'_1)] \cdot \mathbf{x} = [\mathbf{d} \times v'_1] \cdot u'_1.$$

In a similar way, we obtain the following equation for the plane E_2 :

$$[\mathbf{d} \times (v'_2 - u'_2)] \cdot \mathbf{x} = [\mathbf{d} \times v'_2] \cdot u'_2.$$

Since v is the intersection of E_1, E_2 , and the plane through f , we obtain the system of equations $A_{\mathbf{d}}v = b_{\mathbf{d}}$, where $A_{\mathbf{d}}$ is a 3×3 matrix and $b_{\mathbf{d}}$ is a vector. The coefficients of $A_{\mathbf{d}}$ and $b_{\mathbf{d}}$ are polynomials in \mathbf{d} of degree at most one. Hence, we can write v as

$$(4.2) \quad v = A_{\mathbf{d}}^{-1}b_{\mathbf{d}},$$

i.e., each coordinate of v is the quotient of two polynomials in \mathbf{d} , each having degree at most three. This concludes Case 2.

To summarize, using (4.1) and (4.2), each vertex v of the support polygon S on f can be written in the form

$$(4.3) \quad v = \begin{pmatrix} a_1(\mathbf{d})/b_1(\mathbf{d}) \\ a_2(\mathbf{d})/b_2(\mathbf{d}) \\ a_3(\mathbf{d})/b_3(\mathbf{d}) \end{pmatrix},$$

where each of a_1, a_2, a_3, b_1, b_2 , and b_3 is a polynomial in \mathbf{d} of degree at most three.

We are now ready to express the area of the support polygon S as a function of the build direction \mathbf{d} . If the vertices of S are s_0, s_1, \dots, s_{k-1} , then the area of S is equal to

$$(4.4) \quad \sum_{i=0}^{k-1} \Delta(p, s_i, s_{i+1}),$$

where $s_k = s_0$, p is an arbitrary point on f , and $\Delta(p, s_i, s_{i+1})$ is the signed area of the triangle having vertices p, s_i , and s_{i+1} . The sign is positive (resp.

negative), if the points p , s_i , and s_{i+1} form a left-turn (resp. right-turn). (See O'Rourke [O'R98].) Since

$$\Delta(p, s_i, s_{i+1}) = \pm \frac{1}{2} \|(s_i - p) \times (s_{i+1} - p)\|,$$

and since the cross-product on the right-hand side is collinear with \mathbf{n} , we can rewrite the expression in (4.4) as

$$\frac{1}{2} \sum_{i=0}^{k-1} \beta_i [(s_i - p) \times (s_{i+1} - p)] \cdot \mathbf{n},$$

where each β_i has the appropriate value plus or minus one.

In (4.3), we have shown how to express each vertex s_i of the support polygon. Hence, we can write each term $[(s_i - p) \times (s_{i+1} - p)] \cdot \mathbf{n}$ as a quotient of two polynomials in \mathbf{d} , each having degree at most six. In conclusion, we obtain the following expression for the area $A^S(\mathbf{d})$ of the support polygon S :

$$A^S(\mathbf{d}) = \sum_{i=0}^{k-1} \frac{c_i(\mathbf{d})}{d_i(\mathbf{d})},$$

where each of the c_i 's and d_i 's is a polynomial in \mathbf{d} of degree at most six.

Until now, we considered the area $A^S(\mathbf{d})$ of only one support polygon S on one facet f of \mathcal{P} . We have to consider this area for each support polygon on each front facet of \mathcal{P} . Let I be one of the regions in the arrangement on \mathbb{S}^2 , and let K_I be the total number of all vertices of all support polygons on all facets of \mathcal{P} for directions \mathbf{d} in I . Then the total area $A_I(\mathbf{d})$ of all support polygons can be written as

$$(4.5) \quad A_I(\mathbf{d}) = \sum_{i=1}^{K_I} \frac{f_i(\mathbf{d})}{g_i(\mathbf{d})},$$

where each of the f_i 's and g_i 's is a polynomial in \mathbf{d} of degree at most six. Observe that $K_I = O(n^2)$.

The algorithm computes the minimum of the expression (4.5) over all directions $\mathbf{d} \in I$, for each of the $O(n^6)$ regions I . The boundary of each such region I is described by $O(n^3)$ curves on \mathbb{S}^2 . If we denote by $Q(n)$ the time needed to minimize the expression (4.5) within one region I , then we obtain the following result.

THEOREM 4.2. *Let \mathcal{P} be a polyhedron with n facets. A build direction that minimizes the total surface area of \mathcal{P} that is in contact with supports can be computed in $O(n^6 \cdot Q(n))$ time, where $Q(n)$ is the time needed to minimize the expression (4.5).*

It should be clear that the algorithm described in this section is not practical. Therefore, in the next section, we present a heuristic approach for approximating the minimum contact-area.

5. Approximating contact-area of supports for general polyhedra

Let \mathbf{d}^* be a direction that minimizes the contact-area for \mathcal{P} . As seen in the previous section, \mathbf{d}^* appears to be quite difficult to compute. In this section, we give a simple heuristic to compute a build direction $\hat{\mathbf{d}}$ which approximates \mathbf{d}^* . This result is due to Ilinkin *et al.* [IJS⁺04].

Let $CA(\mathbf{d})$ denote the contact-area of \mathcal{P} for a given direction, \mathbf{d} . We first show how to upper-bound the ratio $CA(\hat{\mathbf{d}})/CA(\mathbf{d}^*)$. Let $BFA(\mathbf{d})$ be the total area of the back facets with respect to \mathbf{d} and let \mathbf{d}' be a direction that minimizes the total area of the back facets of \mathcal{P} . Observe that $BFA(\mathbf{d}^*) \leq CA(\mathbf{d}^*)$, since $CA(\mathbf{d}^*)$ also includes possible contact-area on front and parallel facets. Also, by definition of \mathbf{d}' , $BFA(\mathbf{d}') \leq BFA(\mathbf{d}^*)$. Therefore,

$$\frac{CA(\hat{\mathbf{d}})}{CA(\mathbf{d}^*)} \leq \frac{CA(\hat{\mathbf{d}})}{BFA(\mathbf{d}^*)} \leq \frac{CA(\hat{\mathbf{d}})}{BFA(\mathbf{d}')}$$

This result allows us to measure the quality of the approximation provided by our heuristic, for various choices of $\hat{\mathbf{d}}$. In the rest of this section, we show how to compute efficiently $CA(\hat{\mathbf{d}})$ and $BFA(\mathbf{d}')$. We also discuss ways to choose candidate directions $\hat{\mathbf{d}}$ and describe briefly our experimental results. Throughout, our emphasis will be on practical algorithms rather than on the asymptotically most efficient ones.

5.1. Computing contact-area for a fixed direction. The contact-area for a given direction $\hat{\mathbf{d}}$ includes contact-area on front facets, on parallel facets, and on back facets. Handling front and parallel facets is tricky since only parts of these facets may be in contact with supports; handling back facets is straightforward, since they are completely in contact with supports. We describe an algorithm for front facets only; the algorithm for parallel facets is similar.

Our approach is a heuristic based on ray-shooting. Consider a front facet f and let p be a point on f . Point p will be in contact with supports if and only if the ray emanating from p in direction $\hat{\mathbf{d}}$ intersects some other facet of \mathcal{P} . The idea is to pick a set of points in the interior of f and identify those points that will be in contact with supports by shooting rays in direction $\hat{\mathbf{d}}$. Let H_f (resp. M_f) denote the set of rays, originating at points on f , that hit a facet of \mathcal{P} (resp. miss all facets of \mathcal{P}). Then the area of f that is in contact with supports can be approximated as $|H_f|/(|H_f| + |M_f|)$ times the area of f . As the density of the sample points is increased, the accuracy of the approximation improves.

The sample points are selected through a subdivision process, where each facet is subdivided into two triangular patches and placed at the end of a queue of unprocessed patches. Thus, the subdivision proceeds in a breadth-first fashion and all facets are subdivided to the same depth. Each patch is processed by shooting a ray from its centroid in direction $\hat{\mathbf{d}}$ and recording the result. An iteration of the algorithm corresponds to a complete subdivision of the patches. The algorithm terminates after a predefined number d of iterations or when the change in contact-area from one iteration to the next is not significant, e.g., less than 1%. The running time can be shown to be $O(2^d n^2)$.

To verify the accuracy of the heuristic, we also implemented an exact algorithm; however, it is very slow and we use it only for a one-time verification of the heuristic. (This algorithm involves intersecting the projections of pairs of front and back facets and runs in $O(n^3)$ time.) Our experiments on real-world models showed that the heuristic produced answers that were very close to the exact one, and did so in a fraction of the time. (We used $d = 10$ in our experiments.) Details may be found in [IJS⁺04].

5.2. Minimizing the total area of back facets. In Section 4.1, we showed how to minimize the contact-area when \mathcal{P} is convex; in the convex case, this is equivalent to minimizing the total back facet area. It turns out that this algorithm can also be used to minimize the total back facet area for a general polyhedron \mathcal{P} . The algorithm runs in $O(n^2)$ time and uses $O(n)$ space (when topological sweep is used). We describe here a simpler algorithm, which runs in $O(n^2 \log n)$ time and uses $O(n)$ space.

Each facet f of \mathcal{P} defines a great circle C_f on \mathbb{S}^2 , consisting of all points that are at distance $\pi/2$ from the outer unit-normal of f . It can be shown [IJS⁺04] that a direction \mathbf{d}' that minimizes the total back facet area coincides with some vertex of the arrangement \mathcal{A} of these great circles. This vertex can be found, without explicitly computing \mathcal{A} , as follows.

We pick a great circle C_f , compute its intersection with the other great circles, and sort the intersection points (which are vertices of \mathcal{A}) in their circular order around C_f . We pick one of these intersection points and compute the total back facet area for the corresponding direction. We then visit the other intersection points in order and update the total back facet area incrementally. Specifically, suppose that we visit intersection point \mathbf{d}_2 from intersection point \mathbf{d}_1 , and assume that $BFA(\mathbf{d}_1)$ has been computed already. Let $\Delta BFA(\mathbf{d}_1)$ be the total area of the facets that are parallel facets with respect to \mathbf{d}_1 but are back facets with respect to \mathbf{d}_2 . Similarly, let $\Delta BFA(\mathbf{d}_2)$ be the total area of the facets that are parallel facets with respect to \mathbf{d}_2 but are back facets with respect to \mathbf{d}_1 . Then we compute $BFA(\mathbf{d}_2)$ as $BFA(\mathbf{d}_1) + \Delta BFA(\mathbf{d}_1) - \Delta BFA(\mathbf{d}_2)$. The optimum direction \mathbf{d}' is obtained after all great circles C_f have been thus processed.

The running time for each great circle C_f is dominated by the $O(n \log n)$ time to sort the $O(n)$ intersection points. The walk around C_f takes $O(n)$ additional time. Observe that, for each intersection point \mathbf{d}_1 on C_f , the facets that are parallel with respect to \mathbf{d}_1 can be identified from the great circles that intersect at \mathbf{d}_1 . Thus, $\Delta BFA(\mathbf{d}_1)$ can be computed in time proportional to the number of great circles that contain \mathbf{d}_1 , so the time for all intersection points on C_f is $O(n)$. It follows that the running time is $O(n^2 \log n)$. The space used is $O(n)$.

5.3. Choosing candidate directions. We have implemented and tested the following choices for a build direction $\hat{\mathbf{d}}$. In each case, the quality of the heuristic was upper-bounded using the ratio $CA(\hat{\mathbf{d}})/BFA(\mathbf{d}')$, where \mathbf{d}' is computed (just once), as in Section 5.2. A detailed discussion of these choices can be found in [IJS⁺04].

Our choices for $\hat{\mathbf{d}}$ included: (i) a direction which minimizes the total back facet area (i.e., \mathbf{d}'); (ii) a direction which maximizes the total area (or, alternatively, the number) of parallel facets; (iii) the principal components of \mathcal{P} ; and (iv) a direction corresponding to the (inverse image of the) outer normal of that facet of the convex hull of \mathcal{P} which contains facets of \mathcal{P} of maximum total area; intuitively, this direction provides high stability to the part as it is built. We tested these choices on several real models obtained from industry and found savings in contact-area ranging from 9% to 83%, when compared to the contact-area for randomly chosen directions.

In related work, Allen and Dutta [AD95] also describe heuristics for contact-area minimization. They choose a subset of the normals of the convex hull as candidate build directions. For each orientation, they compute the needed supports and contact-area approximately and pick the one that is smallest. No analysis of the quality of the heuristic is given, however. Support optimization has also been

addressed previously by Frank and Fadel [FF94] in the framework of an expert system.

6. Multi-criteria optimization

In the previous sections, we have considered different design criteria that are important to optimize. It is not difficult to see that a single build direction will, in general, not optimize multiple design criteria simultaneously. In this section, we consider the problem of computing a build direction that reconciles multiple measures in a meaningful way. The results we present are due to Majhi *et al.* [MJSS01]. These authors present several algorithms that optimize different combinations of measures. Their algorithms incorporate the results of the previous sections as building blocks. Moreover, their algorithms are modular in the sense that solutions for different combinations of criteria can be derived quickly by combining the appropriate single-criterion algorithms.

We will consider three types of multi-criteria optimization problems. For each type, we will give one specific example. Further applications can be found in [MJSS01].

6.1. Threshold formulation. Let \mathcal{C}_1 and \mathcal{C}_2 be two generic design criteria. Given two designer-specified values ρ_1 and ρ_2 , our goal is to compute a build direction \mathbf{d} such that $\mathcal{C}_1(\mathbf{d}) \leq \rho_1$ and $\mathcal{C}_2(\mathbf{d}) \leq \rho_2$.

Let us assume that $\mathcal{C}_1(\mathbf{d})$ is the maximum error-triangle height $ETH(\mathbf{d})$ for direction \mathbf{d} (see Section 3) and $\mathcal{C}_2(\mathbf{d})$ is the width $W(\mathbf{d})$ of the polyhedron \mathcal{P} in direction \mathbf{d} (see Section 2).

We start by computing a description of all build directions \mathbf{d} for which $ETH(\mathbf{d})$ is less than or equal to ρ_1 . Recall from Section 3 that the error-triangle height $h_f(\mathbf{d})$ for facet f satisfies $h_f(\mathbf{d}) = L \cos \theta_f(\mathbf{d})$. Hence we require that $\theta_f(\mathbf{d}) \geq \theta$ for all facets f , where $\theta := \arccos(\rho_1/L)$.

For each facet f of \mathcal{P} , the set of all directions \mathbf{d} such that $\theta_f(\mathbf{d}) \geq \theta$ can be represented by the complement of the union of two open caps of radius θ and poles \mathbf{n}_f and $-\mathbf{n}_f$, where \mathbf{n}_f is the outer normal of f . This complement, which we call a *band*, consists of two parallel small circles at a distance of $\pi/2 - \theta$ on either side of the great circle defined by \mathbf{n}_f .

Thus, the set of all build directions \mathbf{d} for which $ETH(\mathbf{d}) \leq \rho_1$ is the intersection of these bands, over all facets f of \mathcal{P} . This is a set of regions bounded by arcs of small circles. The total size of the regions is $O(n)$ and they can be computed in $O(n \log n)$ time.

After having computed the intersection of all bands, we proceed as follows. For each region I in the intersection, we want to decide if there is a build direction \mathbf{d} in I for which $W(\mathbf{d}) \leq \rho_2$. This can be done by solving the *constrained width problem* within I , i.e., computing a direction $\mathbf{d} \in I$ that minimizes $W(\mathbf{d})$, and then checking if this width is less than or equal to ρ_2 . Majhi *et al.* [MJSS01] show how this problem can be solved in $O(n^2)$ total time for all regions I .

THEOREM 6.1. *Let \mathcal{P} be a polyhedron with n facets. Given any two real numbers ρ_1 and ρ_2 , we can decide in $O(n^2)$ time, if there exists a build direction \mathbf{d} for which the maximum error-triangle height $ETH(\mathbf{d})$ is less than or equal to ρ_1 and the width $W(\mathbf{d})$ is less than or equal to ρ_2 .*

6.2. Weighted formulation. Again, let \mathcal{C}_1 and \mathcal{C}_2 be two generic measures. In the weighted formulation, we are given two designer-specified weights w_1 and w_2 , reflecting the relative importance of the two criteria, and we want to compute a build direction \mathbf{d} such that $w_1 \cdot \mathcal{C}_1(\mathbf{d}) + w_2 \cdot \mathcal{C}_2(\mathbf{d})$ is minimized.

We again illustrate this type of problem for the case when $\mathcal{C}_1(\mathbf{d})$ is the maximum error-triangle height $ETH(\mathbf{d})$ for direction \mathbf{d} and $\mathcal{C}_2(\mathbf{d})$ is the width $W(\mathbf{d})$ in direction \mathbf{d} .

Let S be the set of directions on \mathbb{S}^2 corresponding to the facet normals (and their inverse images) of \mathcal{P} , as defined in Section 3, and consider the spherical Voronoi diagram $VD(S)$ of S , where distances are measured along \mathbb{S}^2 . Also, consider the graphs G'_u and G'_l on the upper hemisphere \mathbb{S}^2_+ that were defined in Section 2. We define the graph \mathcal{G} on \mathbb{S}^2 to be the overlay of G'_u , G'_l , and their inverse images. Finally, let \mathcal{A} be the arrangement on \mathbb{S}^2 obtained by overlaying $VD(S)$ and \mathcal{G} .

It can be shown that for each face I in the arrangement \mathcal{A} , the expression for the maximum error-triangle height, for directions $\mathbf{d} \in I$, can be written as $ETH(\mathbf{d}) = a_I x + b_I y + c_I z$, where a_I , b_I , and c_I are constants (depending on the face I) and $\mathbf{d} = (x, y, z)$. Similarly, the expression for the width of \mathcal{P} , for directions $\mathbf{d} \in I$, can be written as $W(\mathbf{d}) = a'_I x + b'_I y + c'_I z$, where a'_I , b'_I , and c'_I are again constants that depend on I and $\mathbf{d} = (x, y, z)$. Hence, within the face I of \mathcal{A} , we have to minimize the expression

$$w_1 (a_I x + b_I y + c_I z) + w_2 (a'_I x + b'_I y + c'_I z).$$

Since the face I is bounded by great arcs, its boundary is given by a collection of linear constraints on the variables x , y , and z , together with the constraint $x^2 + y^2 + z^2 = 1$. As shown in Majhi *et al.* [MJS99], the optimization problem for I can be solved using Lagrange Multipliers, in time that is proportional to the number of great arcs on the boundary of I . Since the total size of all faces I sums up to $O(n^2)$, we obtain the following result.

THEOREM 6.2. *Let \mathcal{P} be a polyhedron with n facets. Given any two real numbers w_1 and w_2 , we can compute, in $O(n^2)$ time, a build direction \mathbf{d} for which $w_1 \cdot ETH(\mathbf{d}) + w_2 \cdot W(\mathbf{d})$ is minimum.*

6.3. Sequential formulation. As before, let \mathcal{C}_1 and \mathcal{C}_2 be two generic measures. In the sequential formulation, we want to compute, among all build directions \mathbf{d} that minimize $\mathcal{C}_1(\mathbf{d})$, a direction \mathbf{d} that minimizes $\mathcal{C}_2(\mathbf{d})$.

Assume again that $\mathcal{C}_1(\mathbf{d})$ is the maximum error-triangle height $ETH(\mathbf{d})$ for direction \mathbf{d} , and $\mathcal{C}_2(\mathbf{d})$ is the width $W(\mathbf{d})$ in direction \mathbf{d} .

Consider again the set S of directions on \mathbb{S}^2 corresponding to the facet normals (and their inverse images) of \mathcal{P} . As we saw in Section 3, the set of all directions $\mathbf{d} \in \mathbb{S}^2$ that minimize $ETH(\mathbf{d})$ is equal to the set of poles of all largest empty caps. Moreover, each such pole is the normal of some facet of the convex hull of S . It follows that the directions \mathbf{d} that minimize $ETH(\mathbf{d})$ (there are $O(n)$ of these), can be computed in $O(n \log n)$ time. For each of these directions \mathbf{d} , we have to compute the value of $W(\mathbf{d})$. We can do this by computing and preprocessing the convex hull of \mathcal{P} into the *hierarchical data structure* of Dobkin and Kirkpatrick [DK85]; this takes $O(n \log n)$ time. Using this data structure, we can, given any query direction \mathbf{d} , compute the extreme vertices in that direction, and hence the value of $W(\mathbf{d})$, in $O(\log n)$ time.

THEOREM 6.3. *Let \mathcal{P} be a polyhedron with n facets, and let D be the set of all build directions \mathbf{d} for which $ETH(\mathbf{d})$ is minimum. In $O(n \log n)$ time, we can compute a build direction \mathbf{d} in D that minimizes the width of \mathcal{P} in direction \mathbf{d} .*

7. Protecting facets

Let \mathcal{P} be a polyhedron with n facets, and let f be a fixed facet of \mathcal{P} . In this section, we consider the problem of computing a description of all build directions for which facet f is not in contact with supports. Observe that this problem is equivalent to computing all directions in which f can be translated to infinity such that no collision occurs with the rest of the polyhedron. The approach that we sketch in this section is due to Schwerdt *et al.* [SSJ⁺00]; see also Nurmi and Sack [NS89].

For any point $x \in \mathbb{R}^3$ and for any direction $\mathbf{d} \in \mathbb{S}^2$, we denote by $r_{x\mathbf{d}}$ the ray emanating from x having direction \mathbf{d} . For any facet g of \mathcal{P} and for any point $x \in \mathbb{R}^3$, we define

$$R_{xg} := \{\mathbf{d} \in \mathbb{S}^2 : (r_{x\mathbf{d}} \cap g) \setminus \{x\} \neq \emptyset\}.$$

For any facet g of \mathcal{P} , we define

$$C_{fg} := \bigcup_{x \in f} R_{xg}.$$

Hence, for each direction $\mathbf{d} \in C_{fg}$, facet f is in contact with supports for build direction \mathbf{d} “because of” facet g . That is, there is a point x on f , such that the ray $r_{x\mathbf{d}}$ emanating from x and having direction \mathbf{d} intersects facet g .

LEMMA 7.1. *Ignoring degenerate cases, the complement of the union of the sets C_{fg} , where g ranges over all facets of \mathcal{P} (except for f), is equal to the set of all directions for which facet f is not in contact with supports.*

Lemma 7.1 implies that our problem can be solved in the following way.

Step 1: For each facet g of \mathcal{P} with $g \neq f$, compute a description of the set C_{fg} .

Step 2: Compute the boundary of the union of the sets C_{fg} , $g \neq f$.

We first consider Step 1. It turns out that there is a simple way to compute a set C_{fg} :

LEMMA 7.2. *Let g be a facet of \mathcal{P} . Assume that f and g are disjoint or intersect in a single point. Let*

$$D_{fg} := \{\mathbf{d}_{st} \in \mathbb{S}^2 : s \text{ is a vertex of } f, t \text{ is a vertex of } g, s \neq t\},$$

where \mathbf{d}_{st} is the point on \mathbb{S}^2 having the same direction as the line segment from s to t . Then C_{fg} is the spherical convex hull of the directions in D_{fg} .

Chen and Woo [CW92] present an efficient algorithm for computing the spherical convex hull of a finite set of directions. Observe that, since f and g are triangles, the set D_{fg} in Lemma 7.2 consists of at most nine elements.

Next, we consider Step 2. Observe that after Step 1, we have a collection of at most $n - 1$ spherically convex polygons C_{fg} , each having at most nine edges. Each edge of such a polygon is a great arc. For each such edge e , let K_e be the great circle that contains e . Using an incremental algorithm, we compute the arrangement on \mathbb{S}^2 of the $O(n)$ great circles K_e . (This can be done by adapting the algorithm in,

e.g., de Berg *et al.* [dBvKOS00], for computing the arrangement of lines in \mathbb{R}^2 .) By traversing this arrangement, we then remove all vertices and edges that are inside some set C_{fg} .

THEOREM 7.3. *Let \mathcal{P} be a polyhedron with n triangular facets and let f be a facet of \mathcal{P} . In $O(n^2)$ time and using $O(n^2)$ space, we can compute a description of all build directions for which f is not in contact with supports.*

Implementing the algorithm described above is a highly non-trivial problem. Part of the difficulty comes from degenerate configurations and numerical problems. Observe that if the facets g_1 and g_2 share an edge, then the spherically convex polygons C_{fg_1} and C_{fg_2} will probably also share an edge. In other words, degenerate configurations are very likely to occur. Details about an implementation, based on LEDA [MN99] and its exact rational arithmetic to solve geometric predicates exactly, can be found in Schwerdt *et al.* [SSJJ03] and Schwerdt [Sch01].

The result of Theorem 7.3 is optimal in the sense that there exists a polyhedron \mathcal{P} that contains a facet f such that the set of all build directions for which f is not in contact with supports has $\Omega(n^2)$ connected components.

The algorithm presented in this section can be extended to the case when k facets f_1, f_2, \dots, f_k have to be protected from being in contact with supports. Then, both the running time and the space requirement become $O(k^2 n^2)$.

8. A decomposition-based approach

Given a polyhedron \mathcal{P} with n facets, our goal is to decompose it into a small number of pieces and then build these pieces individually so that the total support requirements (support contact-area or volume) is minimized. Our notion of a decomposition is to cut \mathcal{P} with a plane H normal to some direction \mathbf{d} and build the pieces lying in each halfspace of H in the direction of the normal of H (\mathbf{d} or $-\mathbf{d}$) lying in that halfspace. The quality of the solution depends on \mathbf{d} and, ideally, we would like to find the direction \mathbf{d} which minimizes the support requirements. As a first step towards this challenging goal, we consider the “fixed-direction” version of this problem, where we assume that the direction \mathbf{d} is a given direction. We show how to compute a plane H , perpendicular to \mathbf{d} , which cuts \mathcal{P} into polyhedra \mathcal{P}^+ and \mathcal{P}^- , lying above and below H , respectively, such that the total support contact-area, or support volume, when \mathcal{P}^+ and \mathcal{P}^- are built in directions \mathbf{d} and $-\mathbf{d}$, respectively, is minimized. (See Figure 6.) Since \mathcal{P}^+ and \mathcal{P}^- can consist of several connected components, an additional requirement is that the number of connected components in the optimal decomposition be at most a user-specified integer K . We assume for simplicity that \mathbf{d} is the vertical direction.

This problem is considered in Ilinkin *et al.* [IJM⁺02], where an $O(n \log n)$ -time algorithm is given for a convex polyhedron and it is shown how this solution can be incorporated into an algorithm for a general polyhedron that runs in $O(n^2 \log n)$ time. Experiments on real-world models show significant reductions in support requirements, e.g., 30% to 86% for contact-area. In what follows, we describe these algorithms briefly, focusing on contact-area minimization, since the approach for support volume minimization is similar.

8.1. Decomposing a convex polyhedron to minimize contact-area.

For any position of the plane H , each facet of \mathcal{P} can be classified as *active* or

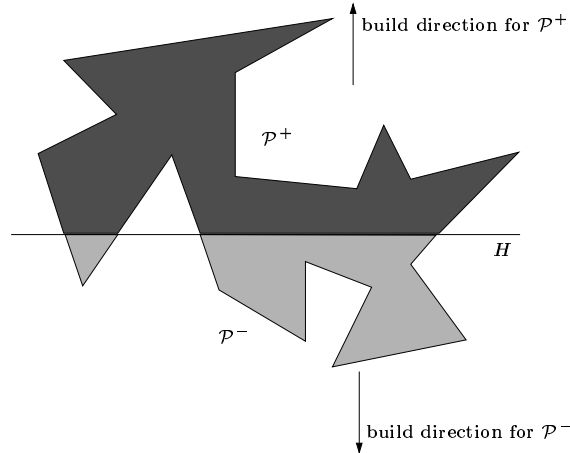


FIGURE 6. The decomposition-based approach, shown in 2D for convenience. The polyhedron \mathcal{P} is decomposed by a plane H into polyhedra \mathcal{P}^+ and \mathcal{P}^- , which are then built in the indicated directions.

inactive, depending on whether or not it is cut by H . An inactive facet is completely contained in either \mathcal{P}^+ or \mathcal{P}^- and its contribution to the contact-area is unaffected by small movements in H . An active facet is contained partially in \mathcal{P}^+ and partially in \mathcal{P}^- and any movement in H affects its contribution to the contact-area. It can be shown that the contribution of an active facet f to the contact-area is a quadratic function $e_f(h) = a_f h^2 + b_f h + c_f$, where h is the height of H above the xy -plane and a_f , b_f , and c_f are constants depending only on the coordinates of the vertices of f . (However, the expressions for these coefficients depend on which edges of f are intersected by H .)

The algorithm begins by sorting the vertices of \mathcal{P} from bottom to top. It then sweeps a horizontal plane H over \mathcal{P} , stopping at each vertex in turn. The total area contributed by the inactive facets (the *inactive-area*) is initialized to the total area of the back facets of \mathcal{P} , and the area contributed by the active facets (the *active-area*) is set to zero; the total contact-area is the sum of these two terms. At the current vertex v , the following actions are taken for each facet f incident to v .

If v is the lowest vertex of f , then f changes from inactive to active. An expression of the form $e_f(h)$ is added to the active-area term and, if f is a back facet of \mathcal{P} , then the area of f is also subtracted from the inactive-area term. (Parallel facets play no role in the convex case and are ignored.) If v is the highest vertex of f , then f changes from active to inactive. An expression of the form $e_f(h)$ is subtracted from the active-area term and, if f is a front facet of \mathcal{P} , then the area of f is added to the inactive-area term (f will henceforth be in \mathcal{P}^- which is built in direction $-\mathbf{d}$, so f will be in contact with supports). If v is the middle vertex of f , then f will continue to be active. The active-area term is updated by subtracting and adding terms of the form $e_f(h)$ (the two terms will be different since H intersects a different edge of f above v than it did below v).

After v is thus processed, a new expression is obtained for the total contact-area. This expression is quadratic in h and does not change for any position of H

between v and its successor. The expression is minimized using standard techniques from calculus to obtain the optimum plane in this range. Once all vertices have been processed, the overall optimum plane is known. The total time to process a vertex is proportional to its degree, so the total time for all vertices is $O(n)$. Including the initial sorting, the overall time is $O(n \log n)$.

8.2. Decomposing general polyhedra. The algorithm of Section 8.1 for convex polyhedra cannot be applied directly to a general polyhedron \mathcal{P} because of the complex nature of supports here (see Section 1.1.3). Nevertheless, it is possible to solve the problem by applying the algorithm for the convex case to a carefully selected set of triangles derived from the facets of \mathcal{P} .

The main idea is to partition each front or back facet of \mathcal{P} into two classes of triangles, called black and gray triangles; one of these classes may be empty. A *black triangle* will always be completely in contact with supports, regardless of the position of the decomposition plane H ; therefore, it always contributes a fixed amount, i.e., its area, to the total contact-area. However, a *gray triangle* will contribute anywhere between zero and its area to the total contact-area, depending on the position of H . Parallel facets are partitioned into three classes of triangles, called black, gray, and white; up to two of these classes may be empty. Black and gray triangles are defined as before; a *white triangle* will never be in contact with supports, regardless of the position of H . It follows that only gray facets need to be considered for support minimization purposes. These are handled using the approach in Section 8.1, as discussed later.

We discuss how to identify black and gray triangles for a front facet f . Suppose that \mathcal{P} is built without decomposition in direction \mathbf{d} . The footprint of the supports, if any, that are in contact with f is a collection of disjoint polygons. Let b be any triangle in a triangulation of the footprint and let g be any triangle in the triangulation of the region that is the complement of the footprint. We claim that b and g satisfy the definition of a black and a gray triangle, respectively.

Suppose that \mathcal{P} is built with decomposition in direction \mathbf{d} . If the decomposition plane H intersects b , then the part b^+ of b above H will be in contact with supports since the portion of \mathcal{P}^+ that is above b^+ is the same as the part of \mathcal{P} that was above b^+ if \mathcal{P} were built without decomposition. The part b^- of b that is below H functions as a back facet in \mathcal{P}^- and will also be in contact with supports. Thus, all of b will be in contact with supports. If H is above b , then $b^+ = \emptyset$ and $b^- = b$; if H is below b , then $b^- = \emptyset$ and $b^+ = b$. In either case, the same argument applies. It follows that b is indeed a black triangle.

Next consider g . If H intersects g , then the part g^- of g that is below H will be in contact with supports since it functions as a back facet in \mathcal{P}^- . Since no part of g is in contact with supports when \mathcal{P} is built in direction \mathbf{d} , it follows that the part g^+ of g , which is in \mathcal{P}^+ and also built in direction \mathbf{d} , will also not be in contact with supports. If H is below or above g , then g is either not in contact or completely in contact with supports. Thus, the contribution of g to the contact-area will be a function of the position of H , so g is indeed a gray triangle. It is easy to check that this contribution has the same form as in the convex case: if g is inactive, it is zero or the area of g ; if g is active, it is a quadratic function of the height of H .

A symmetric discussion applies for back facets. To identify black, gray, and white triangles for a parallel facet f , we build \mathcal{P} without decomposition in directions \mathbf{d} and $-\mathbf{d}$ and identify three types of regions on f : (i) those consisting of points that

are in contact with supports for both directions, (ii) those in contact for exactly one direction, and (iii) those not in contact for either direction. These points define three sets of disjoint polygons that partition f . It is easy to argue as before, that the triangles resulting from triangulating the polygons of types (i)–(iii) are, respectively, black, gray, and white triangles on f .

We can compute the black and gray triangles on the front and back facets simultaneously using the well-known cylindrical decomposition method [Mu194]. This method in essence partitions the space that is outside \mathcal{P} and “in between” its facets into triangular prisms that are oriented in direction \mathbf{d} . From these prisms the footprints of the supports can be inferred easily and the different types of triangles can be computed. The algorithm runs in $O(n^2 \log n)$ time. Parallel facets are handled somewhat similarly. For each parallel facet, the points of types (i)–(iii) are identified by doing a trapezoidal decomposition on a set of line segments obtained by intersecting each non-parallel facet with the plane containing the parallel facet. (Details on these algorithms may be found in [IJM⁺02].)

At this point, we have computed the black, gray, and white triangles for all facets. Recall that only gray triangles are relevant for support minimization. We store the gray triangles from all the facets in a doubly-connected edge list [dBvKOS00] and perform a sweep over them, as in Section 8.1, to compute the optimum plane H . Observe that although the algorithm in Section 8.1 is described for convex polyhedra, it does not really rely on convexity and, in fact, is applicable to any set of triangles.

8.3. Controlling the size of the decomposition. Recall that we assumed that \mathbf{d} is the vertical direction. To ensure that there are at most K connected components in the polyhedra \mathcal{P}^+ and \mathcal{P}^- , a preprocessing step is applied, where the z -axis is partitioned into a sequence \mathcal{I} of disjoint intervals I_1, I_2, \dots , defined by the z -coordinates of the vertices of \mathcal{P} , such that for any position of H within each interval I_j , the number of connected components is the same (say k_j). The above algorithm is then run as before, but the minimization is done only within intervals I_j for which $k_j \leq K$.

The sequence \mathcal{I} is identified via a two-step process. First, a sequence \mathcal{I}' of intervals I'_1, I'_2, \dots , is found such that for any position of H within each interval I'_j , the number of connected components of \mathcal{P}^- is the same. This is done by sweeping a horizontal plane upwards over the sorted sequence of vertices of \mathcal{P} and maintaining the connected components of \mathcal{P}^- using a Union–Find–Makeset data structure; see Cormen *et al.* [CLRS01]. Specifically, the vertices of the connected components are maintained as disjoint sets and when a new vertex is encountered its incident edges are examined and used to merge components together; a count of these components is maintained with the interval defined by the z -coordinates of the current vertex and its successor. The key observation is that during this process, the connected components of \mathcal{P}^- can only merge, never split, so the above data structure suffices. The second step is symmetric and it identifies, via a downward sweep, a sequence \mathcal{I}'' of intervals I''_1, I''_2, \dots , such that for any position of H within each interval I''_j , the number of connected components of \mathcal{P}^+ is the same. The desired sequence \mathcal{I} is inferred by scanning and combining the intervals of \mathcal{I}' and \mathcal{I}'' suitably. This preprocessing adds only $O(n \log n)$ time to the running time of the overall algorithm. (Details may be found in [IJM⁺02].)

We summarize the discussion in this section in the following theorem.

THEOREM 8.1. *Let \mathcal{P} be a polyhedron with n facets. Given a direction \mathbf{d} and an integer K , it is possible to compute a plane H which decomposes \mathcal{P} into at most K connected components such that when the components in the halfspace of H containing \mathbf{d} (resp. $-\mathbf{d}$) are built in directions \mathbf{d} (resp. $-\mathbf{d}$), the total contact-area is minimized. The algorithm runs in $O(n^2 \log n)$ time. A similar result also applies for support volume minimization.*

This result assumes a fixed decomposition direction \mathbf{d} . A natural, but challenging, extension is finding an optimum decomposition over all directions. Some progress has been made on a two-dimensional version of the problem. A simple polygon is a *terrain* if each point on its boundary and interior can be connected to a single edge, called a *base*, by a line segment which is contained completely inside the polygon. Ilinkin *et al.* [IJS02] present algorithms to decompose a simple polygon with a line into two terrains. Their algorithm runs in $O(n \log n)$ time if the two terrains have a common base and in $O(n^2 \log n)$ time otherwise. The motivation for this is that it allows certain classes of polyhedra to be built without supports—specifically, those that are generated by extruding (and possibly scaling) such a decomposable polygon. It is natural to also consider decompositions that minimize the number of terrains. However, Fekéte and Mitchell [FM01] have proved that it is NP-complete to decide if a polygon with holes, or a polyhedron of genus zero, can be decomposed into K terrains, where K is part of the input.

Related problems also arise in the context of casting and injection molding. Rosenbloom and Rappaport [RR94] give an efficient algorithm to decompose a simple polygon into two terrains with a common base. Bose *et al.* [BBvK97] consider the problem of decomposing a polyhedron by a plane and building mold-halves for the two pieces such that the cast polyhedron can be de-molded without obstruction. This is equivalent to deciding if the polyhedron can be decomposed into two polyhedra that are terrains with respect to the common base formed by the dividing plane. Decomposition problems at the interface of Layered Manufacturing and injection molding have been considered recently by McMains [McM02].

9. Computing an optimal hatching direction

In this section, we consider the problem of printing, i.e., hatching, the individual layers. Unlike the other problems considered in this paper, this is essentially a two-dimensional problem since each layer is a polygon. We give an overview of an exact algorithm, due to Schwerdt *et al.* [SSHJ02], and an efficient heuristic, due to Hon *et al.* [HJSS03].

9.1. An exact algorithm. Let P be an n -vertex polygon in the plane, possibly with holes. Let \mathbf{d} be a hatching direction, specified as a unit-vector in the plane. Let $\ell_0(\mathbf{d})$ be the line through the origin in direction \mathbf{d} and let $\mathcal{L}(\mathbf{d})$ be the set of all lines that are parallel to $\ell_0(\mathbf{d})$ and whose distances to $\ell_0(\mathbf{d})$ are integer multiples of δ . Let S_ℓ be the set of line segments of positive length in the intersection of any line $\ell \in \mathcal{L}(\mathbf{d})$ with P and let $H(\mathbf{d}) = \sum_{\ell \in \mathcal{L}(\mathbf{d})} |S_\ell|$. Thus, $H(\mathbf{d})$ is the number of hatching segments for direction \mathbf{d} . Our goal is to find a direction \mathbf{d} such that $H(\mathbf{d})$ is minimized.

Observe that $H(\mathbf{d}) = H(-\mathbf{d})$, so it suffices to consider directions $\mathbf{d} = (d_1, d_2)$ for which $d_2 \geq 0$. The algorithm performs a sweep around the unit-circle, starting from the direction $\mathbf{d} = (-1, 0)$ and going clockwise to $\mathbf{d} = (1, 0)$. During this

sweep, the value of $H(\mathbf{d})$ changes at certain directions \mathbf{d} , called *critical directions*, and we update $H(\mathbf{d})$ at these points. At the end of the sweep, we will have found the optimal hatching direction.

We now state necessary conditions for a direction \mathbf{d} to be critical, i.e., for which the value of $H(\mathbf{d})$ changes. There are two cases:

- \mathbf{d} is such that the subset of lines in $\mathcal{L}(\mathbf{d})$ that intersect P changes. This happens if there is a vertex v of P such that (a) v is extreme in a direction orthogonal to \mathbf{d} , and (b) v lies on a line of $\mathcal{L}(\mathbf{d})$, i.e., the distance between v and $\ell_0(\mathbf{d})$ is an integer multiple of δ .
- \mathbf{d} is such that for some line $\ell \in \mathcal{L}(\mathbf{d})$, the set of segments in S_ℓ changes. This happens if there is a vertex v of P such that (a) both neighbors of v are on the same side of the line through v that is parallel to $\ell_0(\mathbf{d})$, and (b) v lies on a line of $\mathcal{L}(\mathbf{d})$.

Define the set \mathcal{D} to consist of all directions \mathbf{d} for which there is a vertex v of P lying on a line of $\mathcal{L}(\mathbf{d})$. Clearly, \mathcal{D} contains all critical directions.

We begin by computing the set \mathcal{D} , for directions $\mathbf{d} = (d_1, d_2)$ for which $d_2 \geq 0$, and sort these in clockwise order starting from the direction $(-1, 0)$. Let $\mathbf{d}^0 \prec \mathbf{d}^1 \prec \dots \prec \mathbf{d}^{m-1}$ be the m distinct directions in \mathcal{D} . Observe that $H(\mathbf{d}') = H(\mathbf{d}'')$ for any directions \mathbf{d}' and \mathbf{d}'' that are strictly between \mathbf{d}^i and \mathbf{d}^{i+1} .

Next we compute $H(\mathbf{d}^s)$ for some direction \mathbf{d}^s such that $\mathbf{d}^{k-1} \prec \mathbf{d}^s \prec \mathbf{d}^k$. We then consider the elements of \mathcal{D} in the order $\mathbf{d}^k, \mathbf{d}^{k+1}, \dots, \mathbf{d}^{m-1}, \mathbf{d}^0, \dots, \mathbf{d}^{k-1}$. For each such \mathbf{d}^i that is a critical direction, we first compute $H(\mathbf{d}^i)$ from $H(\mathbf{d})$, for $\mathbf{d}^{i-1} \prec \mathbf{d} \prec \mathbf{d}^i$, and then compute $H(\mathbf{d})$ from $H(\mathbf{d}^i)$, for $\mathbf{d}^i \prec \mathbf{d} \prec \mathbf{d}^{i+1}$. After all directions \mathbf{d}^i have been processed in this way, the minimum $H(\mathbf{d})$ found is reported along with the corresponding \mathbf{d} .

Further details on this algorithm can be found in Schwerdt *et al.* [SSHJ02]. We summarize the main result in the following theorem.

THEOREM 9.1. *Let P be an n -vertex polygon, possibly with holes. A direction \mathbf{d} which minimizes the number of hatching segments $H(\mathbf{d})$ can be computed in time $O(Cn \log(Cn))$. Here $C = 1 + L/\delta$, where L is the maximum distance of any vertex of P from the origin.*

Although the algorithm is conceptually simple, care must be taken to handle many special cases. Indeed there are seventy-two different cases to be considered! Nevertheless, the algorithm has been implemented and tested on layers generated from real-world models. Owing to the use of exact arithmetic based on the LEDA number type `real` [MN99], and the dependence on L/δ , the algorithm is fairly slow. However, this algorithm turns out to be crucial in verifying the performance of a fast heuristic which we describe next.

9.2. An efficient heuristic. The idea behind this approach is as follows. Let \mathbf{d} be any hatching direction and let e be any edge of P . The number of times the tool-tip (e.g., the laser in Stereolithography) runs into e is roughly equal to the length of the projection of e normal to \mathbf{d} divided by δ , i.e., $|\mathbf{n}_e \cdot \mathbf{d}|/\delta$, where \mathbf{n}_e is the outer normal to e whose length is the same as that of e . (The error in this bound depends on how far away the first and last lines of $\mathcal{L}(\mathbf{d})$ are from the endpoints of e . The smaller the value of δ , the smaller is the error.) Since each such intersection of the tool-tip with e creates an endpoint of a hatching segment, the total number of hatching segments in direction \mathbf{d} is one-half the sum of the projected lengths of the

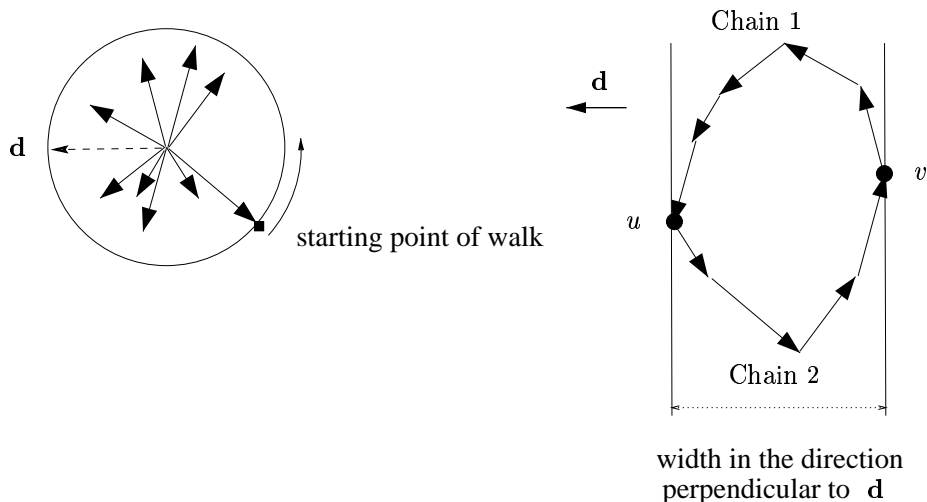


FIGURE 7. A set of vectors and the resulting convex polygon. The sum of the absolute values of the dot products of the vectors with respect to direction \mathbf{d} is twice the width of the convex polygon in the direction perpendicular to \mathbf{d} .

edges divided by δ . Let \mathcal{S} be the set of outer normals of the edges of P , where each normal has the same length as the corresponding edge and begins at the origin. It then follows that an approximate solution to the hatching problem can be obtained by computing a direction \mathbf{d} such that $\sum_{\mathbf{n}_e \in \mathcal{S}} |\mathbf{n}_e \cdot \mathbf{d}|$ is minimized.

The key observation is that $\sum_{\mathbf{n}_e \in \mathcal{S}} |\mathbf{n}_e \cdot \mathbf{d}|$ depends only on the lengths and the orientations of the edges of P and not on how the edges connect to form P . This suggests the possibility of connecting the edges of P in a different order to produce a new polygon Q for which the desired direction can be computed more easily. Indeed, the solution outlined below replaces P with a convex polygon Q and shows that the desired direction \mathbf{d} can be obtained by computing the width of Q .

We first replace all vectors in \mathcal{S} that point in the same direction by a single vector equal to their sum. We then sort these vectors in circular order around the origin and walk through this list to build a chain of vectors as follows; see also Figure 7. Initially, the chain is empty; when we reach the current vector in the walk, we update the chain by adding the current vector so that its tail is at the head of the chain. The chain obtained at the end of the walk will be a closed polygon since P is a closed polygon and each edge of the chain is a vector that is rotated ninety degrees (say, counterclockwise) from an edge of P and has the same length. Moreover, the polygon will be convex since the edges are added in sorted angular order. Denote this convex polygon by Q .

For any direction \mathbf{d} , the boundary of Q can be partitioned into two chains by cutting at the two vertices u and v that are extreme in directions \mathbf{d} and $-\mathbf{d}$, respectively; see Figure 7. The projections of the chains in a direction perpendicular to \mathbf{d} will be non-overlapping (except at u and v). Let ℓ_u and ℓ_v be, respectively, the lines through u and v that are perpendicular to \mathbf{d} . It follows that $\sum_{\mathbf{n}_e \in \mathcal{S}} |\mathbf{n}_e \cdot \mathbf{d}|$ is simply twice the distance between ℓ_u and ℓ_v . Thus, the direction \mathbf{d} which minimizes $\sum_{\mathbf{n}_e \in \mathcal{S}} |\mathbf{n}_e \cdot \mathbf{d}|$ is one which realizes the minimum distance between two parallel

lines that enclose Q , i.e., the *width* of Q . We can, therefore, solve the problem by computing the width of Q , which can be done in $O(n \log n)$ time and using $O(n)$ space; see Houle and Toussaint [HT88]. We note that a similar approach was proposed independently by Sarma [Sar99] in the context of path-planning for milling.

THEOREM 9.2. *Let P be a simple n -vertex polygon, possibly with holes. Let \mathbf{n}_e be the outer normal of edge $e \in P$, where \mathbf{n}_e has the same length as e and begins at the origin. Let \mathcal{S} be the set of such outer normals for all edges of P . Then a direction \mathbf{d} such that $\sum_{\mathbf{n}_e \in \mathcal{S}} |\mathbf{n}_e \cdot \mathbf{d}|$ is minimized can be found in $O(n \log n)$ time and using $O(n)$ space.*

We next describe an alternative algorithm which does not require explicit construction of Q . It works for more general sets of vectors than those derived from the normals of P —a fact that will be useful in some other applications that we mention at the end of the section. The algorithm can also be generalized to higher dimensions.

Henceforth, we assume that \mathcal{S} is a set of n arbitrary vectors in the plane, where each vector begins at the origin. We wish to compute a direction \mathbf{d} such that $\sum_{\mathbf{v} \in \mathcal{S}} |\mathbf{v} \cdot \mathbf{d}|$ is minimized. We pick an arbitrary unit-vector \mathbf{d} as a candidate direction and draw a line perpendicular to \mathbf{d} through the origin. This line cuts the plane into two half-planes. The vectors $\mathbf{v} \in \mathcal{S}$ that are in the same closed half-plane as \mathbf{d} generate a non-negative dot-product with \mathbf{v} . However, those in the complement of the above half-plane generate a negative dot product with \mathbf{d} . We can correct for this by reflecting these vectors through the origin. Let this new set of vectors be $\tilde{\mathcal{S}}$.

All the vectors $\tilde{\mathbf{v}}$ in $\tilde{\mathcal{S}}$ lie in the same closed half-plane as \mathbf{d} . Thus, $\sum_{\mathbf{v} \in \mathcal{S}} |\mathbf{v} \cdot \mathbf{d}|$ reduces to $\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}} \cdot \mathbf{d}$, which can be rewritten as $(\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}) \cdot \mathbf{d}$. If no vector of $\tilde{\mathcal{S}}$ is on the cutting line, we can rotate \mathbf{d} away from $\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}$, thereby decreasing $(\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}) \cdot \mathbf{d}$. We can do this until one of the vectors $\tilde{\mathbf{v}}$ is on the cutting line perpendicular to \mathbf{d} . Any further movement of \mathbf{d} will cause $\tilde{\mathbf{v}}$ to go to the other side of the cutting line. Thereafter, the contribution of (the reflection of) $\tilde{\mathbf{v}}$ will cause $(\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}) \cdot \mathbf{d}$ to increase. Thus, the position of the cutting line that coincides with one of the input vectors must be a local minimum for $(\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}) \cdot \mathbf{d}$.

We update $\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}$ efficiently by visiting the vectors in a circular order, from \mathbf{d} to $-\mathbf{d}$. Specifically, each vector $\tilde{\mathbf{v}}$ has associated with it two regions, separated by the line perpendicular to $\tilde{\mathbf{v}}$. In the walk, whenever we pass this line, the associated vector's contribution to the sum changes sign. If $\tilde{\mathbf{v}}_i$ is the associated vector, we subtract $2\tilde{\mathbf{v}}_i$ from $\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}$: one copy to take it off from the sum, and another copy to insert it back in with a negative sign. At each event, we use the newly updated vector sum to re-calculate $(\sum_{\tilde{\mathbf{v}} \in \tilde{\mathcal{S}}} \tilde{\mathbf{v}}) \cdot \mathbf{d}$. Since the update can be done in $O(1)$ time, we can find the minimum in $O(n)$ time given the circular list of vectors. The total running time is dominated by the time to prepare the list itself, in this case the $O(n \log n)$ time for sorting. This yields the following result, which generalizes Theorem 9.2.

THEOREM 9.3. *Let \mathcal{S} be a set of n arbitrary vectors in the plane, where each vector begins at the origin. A direction \mathbf{d} which minimizes $\sum_{\mathbf{v} \in \mathcal{S}} |\mathbf{v} \cdot \mathbf{d}|$ can be computed in $O(n \log n)$ time and using $O(n)$ space.*

This algorithm has been implemented and tested on layers obtained from real-world models (using a pessimistically large value of 0.1 inches for δ) and the results have been compared to those obtained by the exact algorithm from Section 9.1. It was found that in all cases, the number of hatching segments found by the heuristic was within a small additive constant of the number found by the exact algorithm; moreover, the heuristic was about three to five orders of magnitude faster. A conservative analysis of the heuristic shows that under some reasonable assumptions, the number of hatching segments produced by the heuristic is at most $1 + 3/k$ times the minimum number, for some integer $k \geq 1$. (Details of this analysis and the experiments may be found in [HJSS03].)

We note that it is straightforward to extend the above method to compute an approximation to an optimal hatching direction for all layers of a three-dimensional model: We simply run the algorithm on the set of normals of the edges of the polygons from all layers.

We close by mentioning some other applications of the projection-minimization technique [HJSS03]. Each of these problems can be solved using the second algorithm described above in $O(n \log n)$ time and using $O(n)$ space.

Weighted hatching: The idea here is to assign weights to different edges of the polygons, where the weights reflect the importance of the edge to the form or function of the part. The goal then is to find a direction \mathbf{d} which minimizes the sum of the weights of the edges of P that are hit by the lines in $\mathcal{L}(\mathbf{d})$.

Hatching in two directions: The idea here is to improve the strength of the part by hatching it along two prescribed directions that have some fixed angular relationship to one another. The goal is to find a pair of such directions which minimizes the total number of hatching segments.

Minimizing stair-step error in 2D: The non-zero width δ of the tool-tip implies that the hatched polygon is not an exact replica of the original polygon P ; instead it has a stair-stepped appearance. Analogous to the three-dimensional case discussed in Section 3, one can define an error-triangle on each edge of the polygon and then minimize the sum of the stair-step errors on all edges of P by finding a suitable hatching direction \mathbf{d} .

Finally, we note that Theorem 9.3 can be generalized to $d > 2$ dimensions, using time $O(n^{d-1} \log n)$ (resp. $O(n^{d-1})$) and space $O(n)$ (resp. $O(n^{d-1})$).

10. Final remarks

We have presented an overview of recent work on the application of techniques from computational geometry to several design optimization problems in Layered Manufacturing. Our goal has been to cover the topics in sufficient depth, so that the interested reader can use this as a starting point for further research.

There are several interesting avenues for further research. For instance, can the minimum support volume for general polyhedra be approximated? A challenge in extending the approach presented for contact-area is the absence of a suitable criterion for evaluating the quality of any proposed heuristic for support volume minimization. Another challenging problem is to compute a decomposition of a general polyhedron which is globally optimal, i.e., one which minimizes the contact-area or support volume, when taken over all possible directions $\mathbf{d} \in \mathbb{S}^2$. Similarly, it would be interesting to compute a hatching direction that is optimal over all possible

orientations of the model. Also useful would be other decomposition modalities; for instance, decomposing the model with three planes that are spaced 120° apart. Yet another problem is stair-step error minimization in the presence of adaptive slicing, where the layer thickness is varied dynamically in the vicinity of fine features in the model. The methods presented in this paper assume a fixed layer thickness and it would be interesting to generalize them to the case where the layer thickness lies in some prescribed range. Finally, since the STL format is inherently redundant and error-prone, it would be interesting to consider alternatives (e.g., analytic surfaces such as quadrics, see Farouki and König [FK96]) and the effect of these formats on the algorithms presented here.

We close with a partial list of other interesting aspects of Layered Manufacturing that we have not discussed here. This includes automatic repair of STL files (see Bøhn [Bøh95] and Barequet [Bar97]), methods for support structure elimination through selective thickening of walls of the model (see Allen and Dutta [AD98]), new modeling techniques based on voxels (see Chandru *et al.* [CMP95]), and design issues associated with building a complete software front-end for Layered Manufacturing (see Barequet and Kaplan [BK98]). We refer the reader to Jacobs [Jac92], Kai and Fai [KF97], Chua *et al.* [CLL03], and Stucki *et al.* [SBE95] for more information on these and related topics.

References

- [ABB⁺97] 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** (1997), 61–83.
- [AD95] S. Allen and D. Dutta, *Determination and evaluation of support structures in layered manufacturing*, *Journal of Design and Manufacturing* **5** (1995), 153–162.
- [AD98] S. Allen and D. Dutta, *Wall thickness control in layered manufacturing for surfaces with closed slices*, *Computational Geometry: Theory and Applications* **10** (1998), 223–238.
- [AD00] P. K. Agarwal and P. K. Desikan, *Approximation algorithms for layered manufacturing*, *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, 2000, pp. 528–537.
- [AS96] P. K. Agarwal and M. Sharir, *Efficient randomized algorithms for some geometric optimization problems*, *Discrete & Computational Geometry* **16** (1996), 317–337.
- [Bar97] G. Barequet, *Using geometric hashing to repair CAD models*, *IEEE Computational Science & Engineering* **4** (1997), no. 4, 22–28.
- [Bau75] B. G. Baumgart, *A polyhedron representation for computer vision*, *Proceedings of the AFIPS National Computer Conference*, vol. 44, AFIPS Press, Arlington, Va., 1975, pp. 589–596.
- [BB95] M. Bablani and A. Bagchi, *Quantification of errors in rapid prototyping processes and determination of preferred orientation of parts*, *Transactions of the 23rd North American Manufacturing Research Conference*, 1995.
- [BBvK97] P. Bose, D. Bremner, and M. van Kreveld, *Determining the castability of simple polyhedra*, *Algorithmica* **19** (1997), 84–113.
- [BD90] K. W. Bowyer and C. R. Dyer, *Aspect graphs: An introduction and survey of recent results*, *International Journal of Imaging Systems and Technology* **2** (1990), 315–328.
- [BK98] G. Barequet and Y. Kaplan, *A data front-end for layered manufacturing*, *Computer-Aided Design* **30** (1998), 231–243.
- [BO79] J. L. Bentley and T. A. Ottmann, *Algorithms for reporting and counting geometric intersections*, *IEEE Transactions on Computers* **C-28** (1979), 643–647.
- [Bøh95] J. H. Bøhn, *Removing zero-volume parts from CAD models for layered manufacturing*, *IEEE Computer Graphics and Applications* **15** (1995), no. 6, 27–34.
- [CLL03] C. K. Chua, K. F. Leong, and C. S. Lim, *Rapid prototyping: Principles and applications*, World Scientific, Singapore, 2003.

- [CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.
- [CMP95] V. Chandru, S. Manohar, and C. Prakash, *Voxel-based modeling for layered manufacturing*, IEEE Computer Graphics and Applications **15** (1995), no. 6, 42–47.
- [CW92] L.-L. Chen and T. C. Woo, *Computational geometry on the sphere with application to automated machining*, Journal of Mechanical Design **114** (1992), 288–295.
- [dBvKOS00] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational geometry: Algorithms and applications*, 2nd ed., Springer-Verlag, Berlin, 2000.
- [DK85] D. P. Dobkin and D. G. Kirkpatrick, *A linear algorithm for determining the separation of convex polyhedra*, Journal of Algorithms **6** (1985), 381–392.
- [EG89] H. Edelsbrunner and L. J. Guibas, *Topologically sweeping an arrangement*, Journal of Computer and System Sciences **38** (1989), 165–194.
- [FF94] D. Frank and G. Fadel, *Preferred direction of build for rapid prototyping processes*, Proceedings of the 5th International Conference on Rapid Prototyping, 1994, pp. 191–200.
- [FK96] R. Farouki and T. König, *Computational methods for rapid prototyping of analytic solid models*, Rapid Prototyping Journal **2** (1996), no. 3, 41–48.
- [FM01] S. P. Fekété and J. S. B. Mitchell, *Terrain decomposition and layered manufacturing*, International Journal of Computational Geometry & Applications **11** (2001), 647–668.
- [GH01] B. Gärtner and T. Herrmann, *Computing the width of a point set in 3-space*, Proceedings of the 13th Canadian Conference on Computational Geometry, 2001, pp. 101–103.
- [GO95] A. Gajentaan and M. H. Overmars, *On a class of $O(n^2)$ problems in computational geometry*, Computational Geometry: Theory and Applications **5** (1995), 165–185.
- [HJSS03] M. C. Hon, R. Janardan, J. Schwerdt, and M. Smid, *Minimizing the total projection of a set of vectors, with applications to layered manufacturing*, Computer-Aided Design **35** (2003), 57–68.
- [HT88] M. E. Houle and G. T. Toussaint, *Computing the width of a set*, IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-10** (1988), 761–765.
- [IJM⁺02] I. Ilinkin, R. Janardan, J. Majhi, J. Schwerdt, M. Smid, and R. Sriram, *A decomposition-based approach to layered manufacturing*, Computational Geometry: Theory and Applications **23** (2002), 117–151.
- [IJS02] I. Ilinkin, R. Janardan, and M. Smid, *Terrain polygon decomposition, with application to layered manufacturing*, Proceedings of the 8th International Computing and Combinatorics Conference, Lecture Notes in Computer Science, vol. 2387, Springer-Verlag, 2002, pp. 381–390.
- [IJS⁺04] I. Ilinkin, R. Janardan, M. Smid, E. Johnson, P. Castillo, and J. Schwerdt, *Approximating contact-area of supports in layered manufacturing*, Manuscript, 2004.
- [Jac92] P. F. Jacobs, *Rapid prototyping & manufacturing: Fundamentals of stereolithography*, McGraw-Hill, New York, 1992.
- [KF97] C. C. Kai and L. K. Fai, *Rapid prototyping: Principles and applications in manufacturing*, John Wiley & Sons, 1997.
- [Mat93] J. Matoušek, *Range searching with efficient hierarchical cuttings*, Discrete & Computational Geometry **10** (1993), 157–182.
- [McM02] S. McMains, *Double sided layered manufacturing*, Proceedings of the Japan-USA Symposium on Flexible Automation, 2002, pp. 269–272.
- [MJS⁺99] J. Majhi, R. Janardan, J. Schwerdt, M. Smid, and P. Gupta, *Minimizing support structures and trapped area in two-dimensional layered manufacturing*, Computational Geometry: Theory and Applications **12** (1999), 241–267.
- [MJSG97] J. Majhi, R. Janardan, M. Smid, and P. Gupta, *On some geometric optimization problems in layered manufacturing*, Tech. Report TR-97-002, Department of Computer Science, University of Minnesota, 1997.
- [MJSG99] J. Majhi, R. Janardan, M. Smid, and P. Gupta, *On some geometric optimization problems in layered manufacturing*, Computational Geometry: Theory and Applications **12** (1999), 219–239.

- [MJSS01] J. Majhi, R. Janardan, M. Smid, and J. Schwerdt, *Multi-criteria geometric optimization problems in layered manufacturing*, International Journal of Mathematical Algorithms **2** (2001), 201–225.
- [MN99] K. Mehlhorn and S. Näher, *LEDA: A platform for combinatorial and geometric computing*, Cambridge University Press, Cambridge, U.K., 1999.
- [MS99] S. McMains and C. Séquin, *A coherent sweep plane slicer for layered manufacturing*, Proceedings of the 5th ACM Symposium on Solid Modeling and Applications, 1999, pp. 285–295.
- [Mul94] K. Mulmuley, *Computational geometry: An introduction through randomized algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1994.
- [NS89] O. Nurmi and J.-R. Sack, *Separating a polyhedron by one translation from a set of obstacles*, Proceedings of the 14th International Workshop on Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol. 344, Springer-Verlag, 1989, pp. 202–212.
- [O'R98] J. O'Rourke, *Computational geometry in C*, 2nd ed., Cambridge University Press, Cambridge, UK, 1998.
- [PD90] H. Plantinga and C. R. Dyer, *Visibility, occlusion, and the aspect graph*, International Journal of Computer Vision **5** (1990), 137–160.
- [RR94] D. Rappaport and A. Rosenbloom, *Moldable and castable polygons*, Computational Geometry: Theory and Applications (1994), 219–233.
- [Sar99] S. Sarma, *The crossing function and its application to zig-zag tool paths*, Computer-Aided Design **31** (1999), 881–890.
- [SBE95] P. Stucki, J. Bresenham, and R. Earnshaw, *Computer graphics in rapid prototyping technology*, IEEE Computer Graphics and Applications **15** (1995), no. 6, Edited special issue.
- [Sch01] J. Schwerdt, *Entwurf von Optimierungsalgorithmen für geometrische Probleme im Bereich Rapid Prototyping und Manufacturing*, Ph.D. thesis, Department of Computer Science, University of Magdeburg, Magdeburg, Germany, 2001.
- [SSHJ02] J. Schwerdt, M. Smid, M. C. Hon, and R. Janardan, *Computing an optimal hatching direction in layered manufacturing*, International Journal of Computer Mathematics **79** (2002), 1067–1081.
- [SSJ+00] J. Schwerdt, M. Smid, R. Janardan, E. Johnson, and J. Majhi, *Protecting critical facets in layered manufacturing*, Computational Geometry: Theory and Applications **16** (2000), 187–210.
- [SSJJ03] J. Schwerdt, M. Smid, R. Janardan, and E. Johnson, *Protecting critical facets in layered manufacturing: implementation and experimental results*, Computer-Aided Design **35** (2003), 647–657.
- [SSMJ99] J. Schwerdt, M. Smid, J. Majhi, and R. Janardan, *Computing the width of a three-dimensional point set: an experimental study*, ACM Journal of Experimental Algorithmics **4** (1999), Article 8.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, UNIVERSITY OF MINNESOTA, MINNEAPOLIS, MN 55455, U.S.A.

E-mail address: janardan@cs.umn.edu

SCHOOL OF COMPUTER SCIENCE, CARLETON UNIVERSITY, OTTAWA, ONTARIO, CANADA K1S 5B6

E-mail address: michiel@scs.carleton.ca