On Some Geometric Optimization Problems in Layered Manufacturing ⋆

Jayanth Majhi [1]

Mentor Graphics Corporation, 8005 S.W. Boeckman Road, Wilsonville, OR 97070, U.S.A.

Ravi Janardan [2]

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Michiel Smid [3]

Department of Computer Science, University of Magdeburg, D-39106 Magdeburg, Germany.

Prosenjit Gupta [4]

Delsoft (India) Pvt. Ltd.

Abstract

Efficient geometric algorithms are given for optimization problems arising in layered manufacturing, where a 3D object is built by slicing its CAD model into layers and manufacturing the layers successively. The problems considered include minimizing the stair-step error on the surfaces of the manufactured object under various formulations, minimizing the volume of the so-called support structures used, and minimizing the contact area between the supports and the manufactured object—all of which are factors that affect the speed and accuracy of the process. The stair-step minimization algorithm is valid for any polyhedron, while the support minimization algorithms are applicable only to convex polyhedra. The techniques used to obtain these results include construction and searching of certain arrangements on the sphere, 3D convex hulls, halfplane range searching, and constrained optimization.

Key words: Layered manufacturing, computational geometry, optimization.

1 Introduction

This paper describes efficient algorithms for certain geometric optimization problems arising in layered manufacturing. In *layered manufacturing*, a physical prototype of a 3D object is built from a (virtual) CAD model by orienting and slicing the model with parallel planes and then manufacturing the slices one by one, each on top of the previous one. Layered manufacturing is the basis of an emerging technology called *Rapid Prototyping and Manufacturing* (RP&M). This technology, which is used extensively in the automotive, aerospace, and medical industries, accelerates dramatically the time it takes to bring a product to the market because it allows the designer to create rapidly a physical version of the CAD model (literally on the desktop) and to "feel and touch" it, thereby detecting and correcting flaws in the model early on in the design cycle [15].

Although there are many types of layered manufacturing processes, the basic principle underlying them all is as outlined above. Therefore, for concreteness, we will focus here on just one such method, called *StereoLithography*, which dominates the RP&M market [15]. (In fact, the recent report of the Computational Geometry Task Force explicitly identifies this process as one where geometric techniques could play a significant role [7, page 31].)

1.1 StereoLithography

The input to the StereoLithography process is a surface triangulation of the CAD model in a format called STL. The triangulated model is oriented suitably, sliced by xy-parallel planes, and then built slice by slice in the positive z direction, as follows:

⋆ Portions of this work appear in preliminary form in [17]. The results in this paper, and in a companion paper [18], extend upon and improve the results in [17].

[1] E-mail: jayanth_majhi@mentorg.com. This work was done while the author was at the Department of Computer Science & Engineering at the University of Minnesota. Research supported in part by a Grant–in–Aid of Research from the Graduate School of the University of Minnesota and by NSF grant CCR-9712226.

[2] E-mail: janardan@cs.umn.edu. Research supported in part by a Grant–in–Aid of Research from the Graduate School of the University of Minnesota and by NSF grant CCR-9712226.

[3] E-mail: michiel@isg.cs.uni-magdeburg.de. Part of this work was done while the author was at the Department of Computer Science, King's College London, UK.

[4] E-mail: pgupta@delsoft.com. Part of this work was done while the author was at the Max-Planck-Institut für Informatik, Saarbrücken, Germany, and at the Department of Computer Science & Engineering at the University of Minnesota.

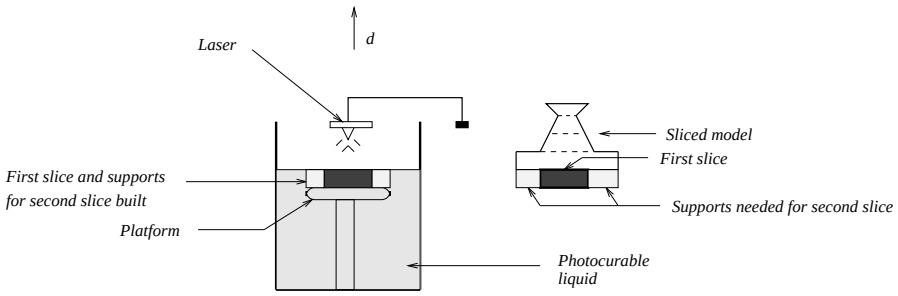

Fig. 1. *The StereoLithograpy apparatus.*

In essence, the StereoLithography Apparatus (SLA) consists of a vat of photocurable liquid resin, a platform, and a laser. Initially, the platform is below the surface of the resin at a depth equal to the slice thickness. The laser traces out the contour of the first slice on the surface and then hatches the interior, which hardens to a depth equal to the slice thickness. In this way, the first slice is created and it rests on the platform. (See Figure 1.)

Next, the platform is lowered by the slice thickness and the just-vacated region is re-coated with resin. The second slice is then built in the same way. Ideally, each slice after the first one should rest in its entirety on the previous one. In general, however, portions of a slice can overhang the previous slice and so additional structures, called *supports*, are needed to hold up the overhangs. Supports are generated automatically during the process itself. For this the CAD model is analyzed beforehand and a description of the supports is generated and merged into the STL file. Supports come in shapes such as wedges, cylinders, and rectangular blocks.

Once the solid has been made, it is postprocessed to remove the supports. Additional postprocessing is often necessary to improve the finish, which has a stair-stepped appearance on certain surfaces due to the non-zero slice thickness used. (See Figure 2, which shows the cross-section of a four-sided "cylinder." The cylinder is normal to the paper and has a uniform cross-section along its entire length.)

1.2 Issues

A key step in layered manufacturing is choosing an orientation for the model, i.e., the *build direction* [15]. Among other things, the build direction affects the quantity of supports used and the surface finish—factors which impact the speed and accuracy of the process. As a simple example, consider building

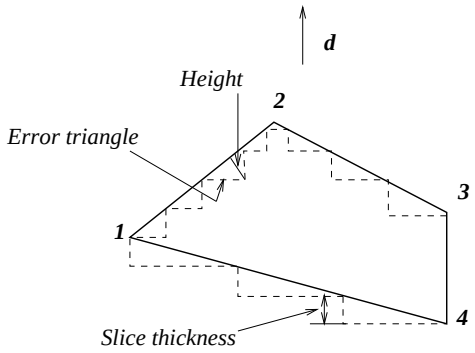

Fig. 2. *Illustrating stair-stepping.*

---

the object in Figure 2: If it is built in the direction **d** indicated in the figure, then the manufactured solid will have a stair-stepped finish and will require supports along the entire length of facet $\overline{14}$, normal to the paper. However, if the build direction is normal to the paper, no supports are needed and there is no stair-stepping on the facets, except possibly on the top and bottom facets (where we mean "top"/"bottom" w.r.t. the build direction). In current systems, the build direction is often chosen by the human operator, based on experience, so that the amount of supports used is "small" and the surface finish is "good". We seek to design computer algorithms that optimize these criteria automatically and lessen the need for human intervention. Indeed, the problems that we consider here are motivated, in part, by discussions with engineers at Stratasys, Inc.—a Minnesota-based world leader in layered manufacturing. (We remark that there is a commercial software package called Bridgeworks for generating supports. The algorithms it uses are proprietary and it is unclear whether they optimize the supports in any way.)

Let us define more formally the parameters of our problem. Throughout the paper, we denote by $\mathcal{P}$ the polyhedral object that we wish to build and by n the number of vertices in $\mathcal{P}$. (Note that there is no loss of generality in assuming a polyhedral model since the input—the STL representation—is polyhedral, even if the original part is not.) We let **d** denote the build direction and, for convenience, imagine it to be vertical so that notions such as "above" and "below" have their usual meaning. Our problem is to find a **d** which minimizes the following three parameters, considered independently (i.e., in isolation from one another):

Stair-step error: Due to the non-zero slice thickness, the manufactured part will have a stair-stepped finish on any facet f that is not parallel to **d**. The degree of stair-stepping on a facet f depends on the angle, $\theta_f(\mathbf{d})$, between the facet normal and **d**, and it can be mitigated by a suitable choice of **d**. In [4],

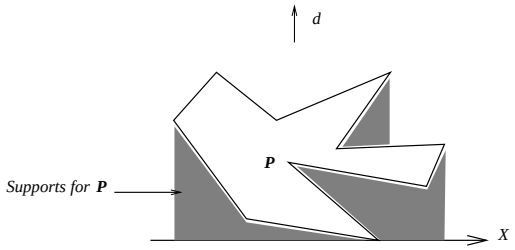

Fig. 3. *Supports for a non-convex polygon.*

---

the notion of an *error-triangle* for a facet is introduced as a way of quantifying stair-stepping (Figure 2). We consider three formulations of stair-step error:

(1) *Minimizing stair-step error:* Here the error on a facet is defined as the height of the error triangle on that facet. The problem is to find a build direction which minimizes the maximum error taken over all the facets.
(2) *Minimizing weighted stair-step error:* From a practical standpoint, not all facets are equally important with respect to surface finish and accuracy. For instance, higher stair-stepping can be tolerated on facets that are not visible or do not touch other parts, while lower stair-stepping is desirable on other more critical facets. This can be handled by assigning weights to the facets (based on their priority). In this case, the error on a facet is defined as the product of its weight and the height of the error triangle. The problem is to minimize the maximum error taken over all the facets.

    We note that the unweighted formulation is a special case of the weighted formulation. However, our algorithm for the unweighted case is direct and more intuitive, and so we have included a separate discussion of this.
(3) *Minimizing sum of stair-step errors:* Here the error on a facet is defined as the sum of the heights of all the error triangles on that facet. The problem is to minimize the total error taken over all the facets.

Volume of supports: The quantity of supports used affects both the building time and the cost. If $\mathcal{P}$ is convex, then the support volume is the volume of the region lying between $\mathcal{P}$ and the platform, i.e., the vertical polyhedral "cylinder" which is bounded below by the platform and above by the facets of $\mathcal{P}$ whose outward normals point downward. If $\mathcal{P}$ is non-convex, then the problem is more complex, since the supports for some facets may actually be attached to other facets instead of to the platform. (Figure 3 illustrates this—in 2D, for convenience.)

Contact area of supports: The amount of $\mathcal{P}$'s surface that is in contact with the supports affects the postprocessing time, since the supports that "stick" to

$\mathcal{P}$ must be removed. If $\mathcal{P}$ is convex, then this is the total area of the downward-facing facets. If $\mathcal{P}$ is non-convex, then this is the total facet area that is in contact with the supports (it includes the areas of all downward-facing facets and portions of certain upward-facing facets.)

1.3 Results

Our main results include:

(1) Simple and practical algorithms for minimizing the maximum stair-step error and the maximum weighted stair-step error of the facets. These algorithms run in time $O(n \log n)$, with the most involved step being merely the computation of the convex hull of a point-set in 3D. We have also implemented and tested these algorithms on real-world STL models obtained from industry. We also give an $O(n^2)$ time algorithm for minimizing the sum of the stair-step errors of the facets. These algorithms work for any polyhedron—even one which is non-convex and has holes.

(2) An $O(n^2)$-time algorithm for computing a build direction $\mathbf{d}$ which minimizes the volume of supports needed by a convex polyhedron $\mathcal{P}$. The algorithm consists of constructing a certain arrangement on the unit-sphere, $\mathbb{S}^2$, whose regions represent directions for which the combinatorial structure of the supports is invariant. We then show how to write the volume formula for a region in a way that allows us to quickly update it in an incremental fashion as we move from region to region. Within each region, we find the best direction using the method of Lagrangian Multipliers.

(3) An $O(n^2)$-time algorithm for minimizing the surface area of a convex polyhedron $\mathcal{P}$ that is in contact with supports. This algorithm involves computing a certain arrangement on $\mathbb{S}^2$, with weighted faces and finding the lightest face. We transform the problem to the plane and solve it using topological sweep [11]. We also give a faster algorithm for the case where $\mathcal{P}$ is built so that an entire facet is in contact with the platform (this is usually the case in practice because it provides more stability to the part). This algorithm runs in roughly $O(n^{4/3})$ time (ignoring polylog factors), and is based on transforming the supports problem to a halfplane range counting problem on weighted points in 2D.

1.4 Prior work

Surprisingly little work has been done by way of efficient and provably optimal geometric algorithms for layered manufacturing. In [3] efficient algorithms are given for deciding if a part can be made by StereoLithography without

using supports. The problem of generating contact-area-optimal supports is considered in [1] and a heuristic yielding an approximate solution is given, but without any analysis of the running time or the quality of the approximation. The issue of support generation is also addressed in [12], in the context of an expert system, while heuristics are presented in [4,9] for improving the accuracy and finish of the part.

1.5 Organization of paper

The rest of the paper is organized as follows: Section 2 describes our algorithms for minimizing the stair step-error under the three formulations and our implementation of two of the algorithms. In Section 3 we present our algorithm for minimizing the volume of supports for a convex polyhedron. In Section 4 we discuss our algorithm for minimizing the contact-area of supports for a convex polyhedron. We conclude in Section 5 with a discussion of further work.

2 Minimizing stair-step error

Recall our problem: *"Given a polyhedron $\mathcal{P}$ with n vertices, find a direction* $\mathbf{d}$ *which minimizes the maximum error-triangle height"*. (Figure 2.) Let L denote the slice thickness and let $h_f(\mathbf{d})$ denote the height of the error-triangle, $t_f(\mathbf{d})$, for facet f. Let $\theta'_f(\mathbf{d})$ be the angle between $\mathbf{d}$ and the normal, $\mathbf{n}_f$, to f, and let $\theta''_f(\mathbf{d})$ be the angle between $\mathbf{d}$ and $-\mathbf{n}_f$. Let $\theta_f(\mathbf{d}) = \min(\theta'_f(\mathbf{d}), \theta''_f(\mathbf{d}))$. It is easy to check that $h_f(\mathbf{d}) = L\cos\theta_f(\mathbf{d})$. We seek a direction $\mathbf{d}$ such that $\max_f h_f(\mathbf{d})$ is minimized, i.e., $\min_f \theta_f(\mathbf{d})$ is maximized.

Consider the set $S = \{\mathbf{n}_f \cap \mathbb{S}^2, -\mathbf{n}_f \cap \mathbb{S}^2 | f \text{ is a facet of } \mathcal{P}\}$. That is, S consists of the points where the facet normals and their negations intersect the unit-sphere $\mathbb{S}^2$. Note that S has $O(n)$ points or *sites*. We wish to find a direction $\mathbf{d}$, i.e., a point $\mathbf{d}$ on $\mathbb{S}^2$, such that the minimum angle between it and the sites is maximized. Define a *cap* on $\mathbb{S}^2$, with *pole* $\mathbf{d}$ and *radius* θ, as the set of all points on $\mathbb{S}^2$ that are at a distance of at most θ from $\mathbf{d}$, as measured along the surface of $\mathbb{S}^2$. Clearly, our problem is equivalent to finding the largest radius cap which does not contain any site in its interior, i.e., a largest empty cap; the pole of this cap is the desired optimal direction. The following properties of caps are easily shown:

(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 the sites in S lie on the same side of $H(C)$. Conversely, every such plane which intersects $\mathbb{S}^2$ corresponds to

an empty cap.

(2) The larger C is, the closer is $H(C)$ to the origin.

(3) A largest empty cap must have at least three sites on its boundary.

By 1 and 2 above, we need to find a plane that is (a) closest to the origin and (b) has all the sites on one side of it. Let $CH(S)$ be the convex hull of S. By 3, it suffices to consider only the facets of $CH(S)$ when searching for a candidate plane. This follows because the plane containing any facet of $CH(S)$ must contain at least three sites and, moreover, all the sites of S lie on one side of this plane; on the other hand, the plane containing three or more co-planar sites that are not all on a facet of $CH(S)$ will have sites on both sides of it.

Therefore, our algorithm is as follows: We first compute the set S and then compute $CH(S)$. For each facet of $CH(S)$, we determine the plane containing the facet and find the one closest to the origin. We then compute the normal from the origin to this closest plane. The desired optimal direction $\mathbf{d}$ is the intersection of this normal and $\mathbb{S}^2$. The overall time is dominated by the $O(n \log n)$ time for the convex hull computation.

Theorem 1 *Let $\mathcal{P}$ be an n-vertex polyhedron. A direction $\mathbf{d}$ which minimizes the maximum error-triangle height can be found in time $O(n \log n)$.* ∎

Remark 2 The algorithm makes no assumptions about $\mathcal{P}$ and hence works for any polyhedron. Moreover, it is simple and practical. In fact, we were able to implement a preliminary version of the algorithm in a matter of a few hours using public-domain software for the convex hull computation. (We used the `Qhull` program [5].) We have tested the algorithm on actual STL datafiles obtained from industry. Figure 4 shows the optimal orientation found by our algorithm for such a test part. For this part, the algorithm took about 12 seconds, excluding the time for graphical output, on an SGI Irix 5 machine.

Remark 3 The algorithm assumes that the slice thickness, L, is fixed. Consider building a rectilinear polyhedron P, i.e., one where any two facets are mutually orthogonal or parallel. If P is built in a direction $\mathbf{d}$ such that the facets are all either normal or parallel to $\mathbf{d}$, then there will be stair-stepping on those facets normal to $\mathbf{d}$ whose distance from the platform is not a multiple of L. (The height of the stair-step on these facets can be arbitrarily close to L.) Our algorithm would reduce this error by building P in a different, and, perhaps, less natural orientation. An interesting problem is to devise algorithms that allow variable slice thicknesses.

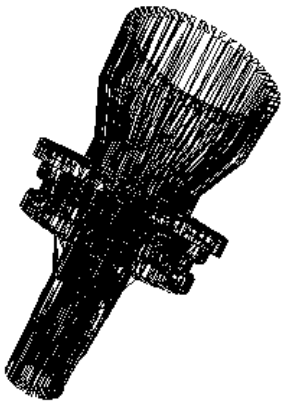

Fig. 4. *Optimal orientation found by our stair-stepping algorithm for a test part—a speedometer component for an automobile. (For clarity, only one-third of the roughly 17,000 facets are displayed here.) The part has been oriented so that the build direction is vertical. Notice that this is quite different from the natural orientation one might expect, where the build direction is parallel (or, perhaps, perpendicular) to the long axis of the part. Indeed, in industry, this part is built along its long axis, usually on its narrow end.*

2.1 Minimizing weighted stair-step error

Recall that each facet f is assigned a positive weight, w_f, and the error on a facet is given by $w_f h_f(\mathbf{d})$ for a build direction $\mathbf{d}$. Since,

$$w_f h_f(\mathbf{d}) = w_f L \max(\mathbf{n}_f \cdot \mathbf{d}, -\mathbf{n}_f \cdot \mathbf{d}) = L \max((w_f \mathbf{n}_f) \cdot \mathbf{d}, (-w_f \mathbf{n}_f) \cdot \mathbf{d}),$$

and L is a constant, the problem is to find a $\mathbf{d}$ such that

$$\max_f \max((w_f \mathbf{n}_f) \cdot \mathbf{d}, (-w_f \mathbf{n}_f) \cdot \mathbf{d})$$

is minimum. Now, $w_f \mathbf{n}_f$ is the unit normal to the facet f scaled by a factor of w_f, and $(w_f \mathbf{n}_f) \cdot \mathbf{d}$ is the (signed length of the) projection of this vector along the direction $\mathbf{d}$. Let the error along direction $\mathbf{d}$ be defined as $error(\mathbf{d}) = \max_f \max((w_f \mathbf{n}_f) \cdot \mathbf{d}, (-w_f \mathbf{n}_f) \cdot \mathbf{d})$, which is simply the maximum of the projections of all the scaled normals (and their negations) along $\mathbf{d}$. So the problem is to find a direction $\mathbf{d^*}$ such that the maximum projection is minimized.

Let S be the set $\{w_f \mathbf{n}_f, -w_f \mathbf{n}_f | f \text{ is a facet of } \mathcal{P}\}$. That is, S consists of the endpoints of the scaled normals and their negations. Let $CH(S)$ be the convex hull of S.

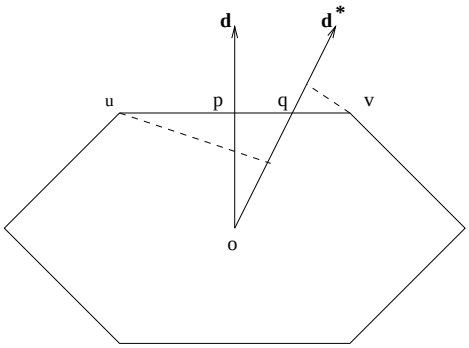

Fig. 5. *The optimal build direction must be normal to one of the facets.*

Lemma 4 *The optimal direction* $\mathbf{d}^*$ *must be normal to one of the facets of* $CH(S)$.

Proof. Assume that the lemma is false. Let the ray from the origin in direction $\mathbf{d}^*$ intersect the facet f of $CH(S)$ at the point q. (See Figure 5, which is shown in 2D for simplicity. Here $f = \overline{uv}$.) Let $\mathbf{d}$ correspond to the direction which is normal to the facet f. Clearly, at least one of the vertices of f, which represents a scaled normal, will project at or beyond q along direction $\mathbf{d}^*$. (This is v in the figure.) Therefore, $error(\mathbf{d}^*) \geq oq$. Consider the direction $\mathbf{d}$. Clearly, $error(\mathbf{d}) = op < oq \leq error(\mathbf{d}^*)$, a contradiction. ∎

Based on the above lemma, we can solve our problem as follows: We compute the set S and its convex hull $CH(S)$. We then find the facet of $CH(S)$ which is closest to the origin. The normal to this facet gives the optimal build direction.

Theorem 5 *Let* $\mathcal{P}$ *be an* n*-vertex polyhedron. A direction* $\mathbf{d}$ *which minimizes the weighted stair-step error taken over all the facets can be found in time* $O(n \log n)$. ∎

Remark 6 We have implemented the weighted algorithm as well. In our implementation, we assigned to each facet a weight equal to its area. (This was simply for convenience. We can easily incorporate other choices of weights in our implementation.) Figure 6 shows the optimal orientation found by our algorithm for the speedometer part. For this part, the algorithm took about 9 seconds, excluding the time for graphical output, on an SGI Irix 5 machine. (The speedup over the unweighted case can be attributed to the fact that the convex hull of S is much smaller in the weighted case.)

Fig. 6. *Optimal orientation found by our weighted stair-stepping algorithm for the speedometer component. The part has been oriented so that the build direction is vertical. Notice again how the build direction is quite different from the natural one, which is along the long axis of the part.*

2.2 Minimizing the sum of stair-step errors

Another criterion that is useful to minimize is the sum of the (weighted) stair-step errors taken over all the facets. Let $m_f(\mathbf{d})$ represent the number of error triangles on the facet f when the part is built in direction $\mathbf{d}$. Then the total weighted stair-step error on facet f is $m_f(\mathbf{d})w_f h_f(\mathbf{d})$. Our problem is to find a $\mathbf{d}$ such that the sum of this over all facets is minimized, i.e.,

$$\min_{\mathbf{d}} \sum_f m_f(\mathbf{d})w_f h_f(\mathbf{d}) = \min_{\mathbf{d}} \sum_f m_f(\mathbf{d})w_f L \max\{\mathbf{n}_f \cdot \mathbf{d}, -\mathbf{n}_f \cdot \mathbf{d}\}.$$

Let $\mathbf{h}_f(\mathbf{d})$ and $\mathbf{l}_f(\mathbf{d})$ be the highest and lowest vertices of f in direction $\mathbf{d}$. Then

$$m_f(\mathbf{d}) = (\mathbf{h}_f(\mathbf{d}) \cdot \mathbf{d} - \mathbf{l}_f(\mathbf{d}) \cdot \mathbf{d})/L.$$

So, the weighted error due to a facet is determined by its normal (or its negation) while the number of error triangles is determined by the highest and lowest points on the facet w.r.t to the build direction. The idea is to construct an arrangement, $\mathcal{A}$, on $\mathbb{S}^2$ with the following property: If R is any face of $\mathcal{A}$, then for all directions $\mathbf{d} \in R$, for each facet of $\mathcal{P}$, the same vertices determine the highest and lowest points and the normal (or its negation) determines the weighted error. We compute $\mathcal{A}$ as the overlay of two arrangements $\mathcal{A}'$ and $\mathcal{A}''$. Faces of $\mathcal{A}'$ are such that for any two directions within a face of $\mathcal{A}'$, the normal (or its negation) determines the weighted error for each facet. Faces of

$\mathcal{A}''$ are such that for any two directions within a face of $\mathcal{A}''$, the same vertices determine the highest and lowest points of each facet.

How do we construct $\mathcal{A}'$? For a build direction $\mathbf{d}$, we call the facets of $\mathcal{P}$ whose outward normals make an angle greater than $\pi/2$ with $\mathbf{d}$ the *back facets* of $\mathcal{P}$; any other facet of $\mathcal{P}$ is a *front facet* w.r.t. $\mathbf{d}$. Note that if f is a back facet w.r.t $\mathbf{d}$, then $\mathbf{n}_f \cdot \mathbf{d} \leq -\mathbf{n}_f \cdot \mathbf{d}$. So, for a back facet, f, $\max\{\mathbf{n}_f \cdot \mathbf{d}, -\mathbf{n}_f \cdot \mathbf{d}\}$ is $-\mathbf{n}_f \cdot \mathbf{d}$, while for a front facet, f, $\max\{\mathbf{n}_f \cdot \mathbf{d}, -\mathbf{n}_f \cdot \mathbf{d}\} = \mathbf{n}_f \cdot \mathbf{d}$. Therefore, the error due to f is determined by $\mathbf{n}_f$ if f is a front facet; otherwise it is determined by $-\mathbf{n}_f$. The set of directions for which $\mathbf{n}_f$ determines the error can be represented by a hemisphere, h_f, whose pole is the point $\mathbf{n}_f$ on $\mathbb{S}^2$. Let g_f be the great circle bounding h_f. Then $\mathcal{A}'$ is the arrangement of g_f's for all $f \in \mathcal{P}$; it has size $O(n^2)$ and it can be computed in $O(n^2)$ time by mapping the great circles to straight lines in the plane via central projection [23] and then using the algorithm in [8].

Next we construct the arrangement, $\mathcal{A}''$ as follows: Recall that the facets of $\mathcal{P}$ are all triangles. Consider a facet $f \in \mathcal{P}$. Let its vertices be u, v and w. For what directions is u the highest point of f? Consider the direction $\mathbf{vu}$. Let h_{vu} be the hemisphere having $\mathbf{vu}$ as its pole. Then, u will be higher than v exactly for directions within h_{vu}. Similarly, u will be higher than w for directions in h_{wu}. Therefore, u will be the highest point of f for directions in $h_{vu} \cap h_{wu}$ and it will be the lowest point for directions in $-(h_{vu} \cap h_{wu})$. Thus, $\mathcal{A}''$ consists of an arrangement of $O(n)$ great circles (one per edge of $\mathcal{P}$). It too can be computed in $O(n^2)$ time.

We compute the desired arrangement, $\mathcal{A}$, as the overlay of $\mathcal{A}'$ and $\mathcal{A}''$ in $O(n^2)$ time. Consider a face $R \in \mathcal{A}$. Let $\mathbf{d} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ be any direction within R. Then for a facet $f \in \mathcal{P}$, $m_f(\mathbf{d}) = (\mathbf{h}_f(\mathbf{d}) \cdot \mathbf{d} - \mathbf{l}_f(\mathbf{d}) \cdot \mathbf{d})/L$, is a linear function of the form $a_f x + b_f y + c_f z$. The error due to f, which is given by $w_f L \max\{\mathbf{n}_f \cdot \mathbf{d}, -\mathbf{n}_f \cdot \mathbf{d}\}$, is also a linear function of the form $a'_f x + b'_f y + c'_f z$. Therefore, the total error due to f is given by a quadratic function and the total error over all the facets is also given by a quadratic function of the same form, namely, $Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz$. (Here a_f–c_f, a'_f–c'_f, and A–F are all constants depending only on f, w_f, L and R.)

Therefore, finding the optimal direction $\mathbf{d} \in R$ involves solving the following constrained optimization problem:

$$\textit{Minimize} \quad f(x, y, z) = Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz$$

$$\begin{aligned} \textit{Subject to} \quad & x^2 + y^2 + z^2 = 1 \\ & a_i x + b_i y + c_i z \geq 0 \text{ for each great arc } r_i \text{ bounding } R, \end{aligned}$$

where a_i–c_i are constants depending on r_i. This problem also arises when we consider support volume minimization in Section 3. Therefore, we defer further

discussion of the problem to Section 3.3, and note here that it can be solved via the method of Lagrangian Multipliers [16] in $O(|R|)$ time, where $|R|$ is the number of edges on R's boundary.

Note that when we move from R to one of its neighboring faces, say R', the optimization problem changes. The new constraints can be set up in $O(|R'|)$ time. But how do we update the objective function? Note that the stair-step error formula changes for only one facet. Specifically, either the weighted error due to the facet changes, or the number of error triangles on that facet changes. Therefore, the objective function can be updated in $O(1)$ time. We visit the faces of $\mathcal{A}$ via a depth-first search of its dual graph and update and solve the optimization problem at each face. Clearly, the total time over all the faces is $\sum_{R \in \mathcal{A}} O(|R|) = O(n^2)$.

Theorem 7 *Let $\mathcal{P}$ be an n-vertex polyhedron. A direction $\mathbf{d}$ which minimizes the sum of stair-step errors can be found in time $O(n^2)$.* ∎

3 Minimum-volume supports for a convex polyhedron

Let us call the vertex v of $\mathcal{P}$ that is farthest away in direction $-\mathbf{d}$ the *extreme vertex* of $\mathcal{P}$. Thus, when $\mathcal{P}$ is built in direction $\mathbf{d}$, v rests on the platform and the facets requiring support are the back facets. The support volume is the volume of the polyhedron which is bounded below by the platform, above by the back facets, and on the sides by vertical facets that contain the edges on the boundary of the union of the back facets. (See Figure 7.)

Our approach consists of partitioning $\mathbb{S}^2$, the set of all directions in 3-space, into $O(n^2)$ regions, R, such that the back facets and the extreme vertex are the same for all directions $\mathbf{d} \in R$. This partition can be represented as an arrangement $\mathcal{A}$ on the unit-sphere $\mathbb{S}^2$. We generate a formula for the total support volume w.r.t. any $\mathbf{d} \in R$ and set up an optimization problem for finding the build direction which minimizes the total support volume within R. We solve this problem, in time $O(|R|)$, by using the Lagrangian Multipliers method and by exploiting the convexity of R. This implies that the total time for all regions is $O(n^2)$.

However, we must also set up the volume formula within each region R. Doing this in the straightforward way takes $O(n)$ time per region, hence $O(n^3)$ time overall. We circumvent this by visiting the regions of the arrangement in a certain order and updating the formula incrementally. For this, we rewrite the formula as two parts—one based mainly on the extreme vertex v for R and the other based on the back facets. We then show that after doing an $O(n^2)$-time precomputation, the formula in this new form can be updated incrementally

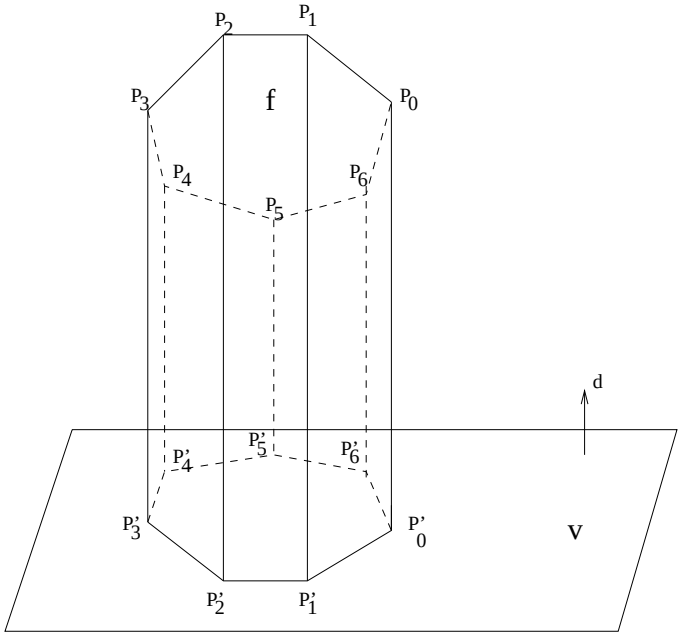

Fig. 7. *A back facet f of $\mathcal{P}$ and the corresponding support polyhedron $\mathcal{P}_f$. (The entire polyhedron $\mathcal{P}$ is itself not shown.)*

---

in $O(n^2)$ total time as we visit the regions of $\mathcal{A}$.

Here is our approach in more detail: Following McKenna and Seidel [21], we take for each facet $f \in \mathcal{P}$, the plane t_f which is parallel to f and passes through the origin. The planes t_f partition 3-space into unbounded polyhedral regions called *cones*, each with its apex at the origin and such that for all directions within a cone the set of back facets is the same. We can represent these cones on $\mathbb{S}^2$ as the arrangement, $\mathcal{A}'$, of the great circles $t_f \cap \mathbb{S}^2$, i.e., each cone is in 1-1–correspondence with a region of $\mathcal{A}'$. Note that $\mathcal{A}'$ is composed of arcs of great circles, has size $O(n^2)$, and can be computed in time $O(n^2)$ using the algorithm given in [8]. It is obtained in a canonical form, where the edges incident at each vertex are in sorted order around the vertex. This allows the boundary of each face of $\mathcal{A}$ to be retrieved in time linear in its size.

Next using the algorithm of Bose *et al.* [6], we compute a second arrangement, $\mathcal{A}''$, of great circle arcs on $\mathbb{S}^2$ such that all directions within a region of $\mathcal{A}''$ have the same extreme vertex. As shown in [6], the directions for which a particular vertex $v \in \mathcal{P}$ is the extreme vertex can be obtained as follows: Translate $\mathcal{P}$ such that v is at the origin. For each neighbor vertex w of v, let t_{vw} be the plane normal to vw and passing through v. Let T_{vw} be the closed halfspace bounded by t_{vw} such that it contains vw. Then the set of directions for which v is the extreme vertex is determined by the intersection of all the T_{vw}'s. So

we can compute a region r of $\mathcal{A}''$ in time $O(|r| \log |r|)$ and represent it on $\mathbb{S}^2$ as a polygon composed of arcs of great circles. Note that $|r|$ is equal to the number of vertices w that are neighbors of v. Thus we can compute all the regions of $\mathcal{A}''$ in time $O(n \log n)$ and, by sorting the edges at all the vertices in additional $O(n \log n)$ time, we can obtain it in the above canonical form.

The desired arrangement, $\mathcal{A}$, is the overlay of $\mathcal{A}'$ (which is an arrangement of great circles) and $\mathcal{A}''$ (which is an arrangement of great arcs). It can be obtained in canonical form in $O(n^2)$ time as follows: It is convenient to think of the great circles of $\mathcal{A}'$ as being colored red and the great arcs of $\mathcal{A}''$ as being colored blue. We extend each blue arc of $\mathcal{A}''$ to the corresponding great circle, where the extensions are colored light blue. We compute the arrangement, $\mathcal{B}$, of the set of $O(n)$ great circles from $\mathcal{A}'$ and $\mathcal{A}''$. $\mathcal{B}$ is obtained in canonical form and its edges are colored red or blue or light blue. We can obtain $\mathcal{A}$ from $\mathcal{B}$ by deleting the light blue edges. Towards this end, we do a depth-first search of $\mathcal{B}$. During the search, whenever we traverse a light blue edge for the second (i.e., last) time, we delete it. In the resulting arrangement, all vertices will have even degree greater than or equal to zero. Vertices of degree zero are the ones in $\mathcal{B}$ that had only light blue edges incident to them; these vertices are deleted. Vertices of degree two must have two red or two blue edges incident to them; these vertices are deleted and the incident edges replaced by a single edge of the same color. Vertices of higher degree are left alone. It is clear that the resulting arrangement $\mathcal{A}$ is indeed in canonical form and that the total time to construct it is $O(n^2)$.

Note that we also need to know for each region $R \in \mathcal{A}$ the corresponding set of back facets and the extreme vertex. Rather than store this explicitly, which would take $O(n^3)$ total space, we compute them incrementally when we update the volume formula, as described in Section 3.4.

3.1 Generating the volume formula

Given a region $R \in \mathcal{A}$, we determine a formula for the volume, $V(R)$, of the supports required by $\mathcal{P}$ for any direction $\mathbf{d} \in R$. Let $\mathbf{d} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ be any unit-vector within R. Let $\mathbf{v}$ be the extreme vertex for R. Consider one of the back facets, say, f. Let the vertices of the facet be $P_0, P_1, \ldots, P_{m-1}$, in counterclockwise order looking from outside the polyhedron. Let this facet project onto convex polygon $P'_0, P'_1, \ldots, P'_{m-1}$ on the plane which passes through $\mathbf{v}$ and is normal to direction $\mathbf{d}$. The volume of the supports needed by f is then the volume of the polyhedron $\mathcal{P}_f$ shown in Figure 7. We have,

$$\mathbf{P}_i - \mathbf{P}'_i = k_i \mathbf{d},$$

and

$$(\mathbf{P}'_i - \mathbf{v}) \cdot \mathbf{d} = 0, \tag{1}$$

where k_i is given by

$$k_i = (\mathbf{P}_i - \mathbf{v}) \cdot \mathbf{d}.$$

(We denote the position vector of a vertex P_i in boldface, as $\mathbf{P}_i$.) Let the facets of $\mathcal{P}_f$ be $S_0, S_1, \ldots, S_{m+1}$, where S_0 is the top facet, S_1 is the bottom facet, and S_i $(2 \leq i \leq m+1)$ is a side facet, i.e., $S_i = P_{i-2}P_{i-1}P'_{i-1}P'_{i-2}$. Let $\mathbf{Q}_j$ be any point on S_j, and let $\mathbf{N}_j$ be a unit outward-normal vector for S_j (we mean "outward" w.r.t. $\mathcal{P}_f$). Then, using the formula in [14], the volume, $V_f(R)$, of $\mathcal{P}_f$ is given by

$$V_f(R) = \frac{1}{3}\sum_j (\mathbf{Q}_j \cdot \mathbf{N}_j) Area(S_j).$$

Let $V_{if}(R)$ be the volume contributed by S_i to $V_f(R)$, $0 \leq i \leq m+1$. Then,

$$V_{0f}(R) = \frac{1}{3}(\mathbf{P}_0 \cdot (-\mathbf{n}_f))(\frac{1}{2}(\mathbf{n}_f \cdot (\sum_{i=0}^{m-1} \mathbf{P}_i \times \mathbf{P}_{i+1}))),$$

where $-\mathbf{n}_f$ is the unit outward-normal for facet f ("outward" w.r.t. $\mathcal{P}_f$). (For convenience, we take $P_m = P_0$ and $P'_m = P'_0$.) We call $\sum_{i=0}^{m-1} \mathbf{P}_i \times \mathbf{P}_{i+1}$ the *area term* and denote it by $\mathbf{\Delta}_f$. Thus,

$$V_{0f}(R) = -\frac{1}{6}(\mathbf{P}_0 \cdot \mathbf{n}_f)(\mathbf{n}_f \cdot \mathbf{\Delta}_f).$$

Next, we have

$$V_{1f}(R) = \frac{1}{3}(\mathbf{P}'_0 \cdot (-\mathbf{d}))(\frac{1}{2}(-\mathbf{d} \cdot (\sum_{i=0}^{m-1} \mathbf{P}'_i \times \mathbf{P}'_{i+1}))).$$

Now $\mathbf{P}'_0 \cdot \mathbf{d} = \mathbf{v} \cdot \mathbf{d}$ (from equation (1)) and

$$\begin{aligned}
\mathbf{d} \cdot (\mathbf{P}'_i \times \mathbf{P}'_{i+1}) &= \mathbf{d} \cdot ((\mathbf{P}_i - k_i\mathbf{d}) \times (\mathbf{P}_{i+1} - k_{i+1}\mathbf{d})) \\
&= \mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i+1} - k_{i+1}\mathbf{P}_i \times \mathbf{d} + k_i\mathbf{P}_{i+1} \times \mathbf{d}) \\
&= \mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i+1}).
\end{aligned}$$

Therefore,

$$V_{1f}(R) = \frac{1}{6}(\mathbf{d}\cdot\mathbf{v})(\mathbf{d}\cdot(\sum_{i=0}^{m-1}\mathbf{P}_i\times\mathbf{P}_{i+1})),$$

which can be written as

$$V_{1f}(R) = \frac{1}{6}(\mathbf{d}\cdot\mathbf{v})(\mathbf{d}\cdot\mathbf{\Delta}_f).$$

Recall that $S_i, (2\le i\le m+1)$ is the quadrilateral formed by $P_{i-2}, P_{i-1}, P'_{i-1}$, and P'_{i-2}. The unit outward-normal to S_i is $\mathbf{n}_i = \frac{\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})}{|\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})|}$. The volume, $V_{if}(R)$, contributed by S_i is given by

$$\begin{aligned}&V_{if}(R) = \\ &\frac{1}{3}(\mathbf{P}_{i-2}\cdot\mathbf{n}_i)\\ &(\frac{1}{2}(\mathbf{n}_i\cdot(\mathbf{P}_{i-2}\times\mathbf{P}_{i-1}+\mathbf{P}_{i-1}\times\mathbf{P}'_{i-1}+\mathbf{P}'_{i-1}\times\mathbf{P}'_{i-2}+\mathbf{P}'_{i-2}\times\mathbf{P}_{i-2}))).\end{aligned}$$

Now,

$$\begin{aligned}\mathbf{P}_{i-2}\cdot\mathbf{n}_i &= \mathbf{P}_{i-2}\cdot\frac{\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})}{|\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})|}\\ &= -\frac{\mathbf{d}\cdot(\mathbf{P}_{i-2}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2}))}{|\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})|}\\ &= -\frac{\mathbf{d}\cdot(\mathbf{P}_{i-2}\times\mathbf{P}_{i-1})}{|\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})|}.\end{aligned}$$

Also,

$$\begin{aligned}\mathbf{P}_{i-1}\times\mathbf{P}'_{i-1} &= \mathbf{P}_{i-1}\times(\mathbf{P}_{i-1}-k_{i-1}\mathbf{d}) = k_{i-1}(\mathbf{d}\times\mathbf{P}_{i-1})\\ \mathbf{P}'_{i-1}\times\mathbf{P}'_{i-2} &= (\mathbf{P}_{i-1}-k_{i-1}\mathbf{d})\times(\mathbf{P}_{i-2}-k_{i-2}\mathbf{d})\\ &= \mathbf{P}_{i-1}\times\mathbf{P}_{i-2}+k_{i-2}\mathbf{d}\times\mathbf{P}_{i-1}-k_{i-1}\mathbf{d}\times\mathbf{P}_{i-2}\\ \mathbf{P}'_{i-2}\times\mathbf{P}_{i-2} &= (\mathbf{P}_{i-2}-k_{i-2}\mathbf{d})\times\mathbf{P}_{i-2} = -k_{i-2}\mathbf{d}\times\mathbf{P}_{i-2}.\end{aligned}$$

So we can write,

$$\begin{aligned}V_{if}(R) = \frac{1}{3}(&-\frac{\mathbf{d}\cdot(\mathbf{P}_{i-2}\times\mathbf{P}_{i-1})}{|\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})|})\\ &(\frac{\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})}{|\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})|}\cdot\frac{k_{i-2}+k_{i-1}}{2}(\mathbf{d}\times(\mathbf{P}_{i-1}-\mathbf{P}_{i-2})))\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{6}(\mathbf{d} \cdot (\mathbf{P}_{i-1} \times \mathbf{P}_{i-2}))(k_{i-2} + k_{i-1}) \\
&= \frac{1}{6}(\mathbf{d} \cdot (\mathbf{P}_{i-1} \times \mathbf{P}_{i-2}))(\mathbf{d} \cdot (\mathbf{P}_{i-2} + \mathbf{P}_{i-1} - 2\mathbf{v})) \\
&= \frac{1}{6}(\mathbf{d} \cdot (\mathbf{P}_{i-1} \times \mathbf{P}_{i-2}))(\mathbf{d} \cdot (\mathbf{P}_{i-2} + \mathbf{P}_{i-1})) \\
&\quad + \frac{1}{3}(\mathbf{d} \cdot (\mathbf{P}_{i-2} \times \mathbf{P}_{i-1}))(\mathbf{d} \cdot \mathbf{v}).
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
V_f(R) &= \sum_{i=0}^{m+1} V_{if}(R) \\
&= V_{0f}(R) + V_{1f}(R) + \sum_{i=2}^{m+1} V_{if}(R) \\
&= -\frac{1}{6}(\mathbf{P}_0 \cdot \mathbf{n}_f)(\mathbf{n}_f \cdot \boldsymbol{\Delta}_f) + \frac{1}{6}(\mathbf{d} \cdot \mathbf{v})(\mathbf{d} \cdot \boldsymbol{\Delta}_f) + \\
&\quad \frac{1}{6}\sum_{i=1}^{m}(\mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i-1}))(\mathbf{d} \cdot (\mathbf{P}_i + \mathbf{P}_{i-1})) + \frac{1}{3}(\mathbf{d} \cdot \boldsymbol{\Delta}_f)(\mathbf{d} \cdot \mathbf{v}) \\
&= -\frac{1}{6}(\mathbf{P}_0 \cdot \mathbf{n}_f)(\mathbf{n}_f \cdot \boldsymbol{\Delta}_f) + \frac{1}{6}\sum_{i=1}^{m}(\mathbf{d} \cdot (\mathbf{P}_i \times \mathbf{P}_{i-1}))(\mathbf{d} \cdot (\mathbf{P}_i + \mathbf{P}_{i-1})) + \\
&\quad \frac{1}{2}(\mathbf{d} \cdot \boldsymbol{\Delta}_f)(\mathbf{d} \cdot \mathbf{v}) \\
&= V'_f(R) + \frac{1}{2}(\mathbf{d} \cdot \boldsymbol{\Delta}_f)(\mathbf{d} \cdot \mathbf{v}),
\end{aligned}
$$

where $V'_f(R)$ denotes the part of $V_f(R)$ which is independent of $\mathbf{v}$. As we will see in Section 3.4, being able to decompose the formula in this way is crucial to the running time of the algorithm. Therefore, the total support volume, $V(R) = \sum_f V_f(R)$, associated with region $R \in \mathcal{A}$ is:

$$
\begin{aligned}
V(R) &= \sum_f V'_f(R) + \frac{1}{2}\sum_f (\mathbf{d} \cdot \boldsymbol{\Delta}_f)(\mathbf{d} \cdot \mathbf{v}) \\
&= \sum_f V'_f(R) + \frac{1}{2}(\mathbf{d} \cdot \sum_f \boldsymbol{\Delta}_f)(\mathbf{d} \cdot \mathbf{v}) \\
&= \sum_f V'_f(R) + \frac{1}{2}(\mathbf{d} \cdot \boldsymbol{\Delta}(R))(\mathbf{d} \cdot \mathbf{v}),
\end{aligned}
$$

where $\boldsymbol{\Delta}(R) = \sum_f \boldsymbol{\Delta}_f$ is the total area term for all the back facets associated with R.

3.2 The optimization problem

Let $\mathbf{d} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$. When we expand $V(R)$ we get an expression of the form $Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz + G$, where $A, B, \ldots, G$ are constants depending only on the $\mathbf{P}_i$'s of the different back facets of R and on the extreme vertex v of R.

Let $\mathbf{d}_R \in \mathbb{S}^2$ be a given point in R's interior. ($\mathbf{d}_R$ is computed when $\mathcal{A}$ is constructed.) For any great arc a bounding R, let $h(a)$ be the plane containing the corresponding great circle and let $\mathbf{n}_{h(a)}$ be a unit-normal for it. Note that $\mathbf{d}$ is in the interior of R if and only if $\mathbf{d} \cdot \mathbf{n}_{h(a)}$ and $\mathbf{d}_R \cdot \mathbf{n}_{h(a)}$ are both positive or both negative for each great arc a bounding R. Thus our optimization problem within each region, R, is:

Minimize $\quad f(x, y, z) = Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz$

Subject to $\quad x^2 + y^2 + z^2 = 1 \quad$ (*Sphere Constraint.*)
$\mathbf{d} \cdot \mathbf{n}_{h(a)} \geq 0$ (resp. ≤ 0) if $\mathbf{d}_R \cdot \mathbf{n}_{h(a)} \geq 0$ (resp. ≤ 0) for each great arc a bounding R. (*Plane Constraints.*)

3.3 Solving the optimization problem

We use the method of Lagrangian Multipliers. We proceed in three stages: (i) We first keep only the sphere constraint active and find the extreme points (i.e., minimum or maximum) over all of $\mathbb{S}^2$. (ii) Next, we take some arc a bounding R and make the corresponding plane constraint active as well. This gives extreme points lying on a's great circle. We repeat this for each great arc a bounding R. (iii) Finally, we consider arcs a and a' meeting at a vertex v of R and make the corresponding plane constraints active—thus making v an extreme point. Note that it is not necessary to make more than two plane constraints active since there is no point of R that is common to three great circles.

All plane constraints are inactive: The Lagrangian is $L(x, y, z, \lambda) = f(x, y, z) + \lambda(1 - x^2 - y^2 - z^2)$, for some parameter λ. The partial derivatives of L, w.r.t. each of x, y, and z, must be zero at an extreme (i.e., minimum or maximum) point. This yields three linear equations in x, y, and z. The values of λ for which these three equations have non-trivial solutions can be found

by solving a cubic equation in λ, given by:

$$\begin{vmatrix} 2A-2\lambda & D & F \\ D & 2B-2\lambda & E \\ F & E & 2C-2\lambda \end{vmatrix} = 0.$$

For each such real-valued λ (there are at most three of them) we solve for x, y, and z, using any two of the three linear equations (the remaining one will depend on the two chosen) and the sphere constraint. This will yield (i) two antipodal points on $\mathbb{S}^2$, or (ii) a great circle (if the three equations are the same but not identically zero), or (iii) all of $\mathbb{S}^2$ (if the three equations are identically zero). We can ignore cases (ii) and (iii) since, anyway, we will be covering them in the cases below. If case (i) holds then we check if either of the two points lies in R (by checking the plane constraints) and, if so, we use this point as a candidate for the minimum value of $f(x, y, z)$.

One plane constraint is active: Let this plane be defined by $ax+by+cz = 0$. Then the Lagrangian is $L(x, y, z, \lambda_1, \lambda_2) = f(x, y, z)+\lambda_1(1-x^2-y^2-z^2)+\lambda_2(ax + by + cz)$, for some parameters λ_1 and λ_2. Setting partial derivatives to zero gives three linear equations in x, y, z, and λ_2. Using these equations and the equation $ax + by + cz = 0$ we can compute the values of λ_1 that yield non-trivial solutions, this time by solving a quadratic equation in λ_1, as given by:

$$\begin{vmatrix} 2A-2\lambda_1 & D & F & a \\ D & 2B-2\lambda_1 & E & b \\ F & E & 2C-2\lambda_1 & c \\ a & b & c & 0 \end{vmatrix} = 0.$$

We can now eliminate λ_2 using one of the linear equations. Using the sphere constraints, the constraint $ax + by + cz = 0$, and any one of the remaining linear equations, we proceed to compute the extreme points and check for feasibility.

Two plane constraints are active: In this case we only need to consider the vertices of R, since these are the only points that are common to two great circles bounding R and are in the feasible region.

Analysis: The cubic and quadratic equations that arise can be solved in $O(1)$ time. Thus, the optimization problem for R takes time $O(|R|)$. Summed over all regions, this is $O(n^2)$.

Remark 8 In general, the Lagrangian Multipliers method runs in time which is exponential in the number of constraints, as we have to make each subset of the $|R|$ constraints active, in turn. However, in our problem R is a convex region so it is not necessary to make more than two plane constraints active in turn (since there is no point of R that is common to three great circles bounding R). Furthermore, we only need to consider $O(|R|)$ pairs of plane constraints that correspond to neighboring edges of R. (Note that if R is a non-convex region then we also have to consider pairs of plane constraints that do not necessarily correspond to neighboring edges of R, and there are $O(|R|^2)$ such pairs.)

3.4 Updating the volume formula incrementally

We precompute $\mathbf{\Delta}_f$ for each facet $f \in \mathcal{P}$. Then, we pick an initial region, $R_0 \in \mathcal{A}$, and compute its back facets, its extreme vertex, and $V(R_0)$ and $\mathbf{\Delta}(R_0)$. Clearly, all this can be done in $O(n)$ time. Next, we compute the dual graph, $\hat{\mathcal{A}}$, of $\mathcal{A}$, by placing a vertex in each region of $\mathcal{A}$ and joining two vertices by an edge if the corresponding regions share an edge. We then visit the regions of $\mathcal{A}$ in the order given by a depth-first search of $\hat{\mathcal{A}}$ which starts at the vertex corresponding to R_0. Suppose that the search moves from region R to region R'. There are three cases:

(1) A facet f which was a back facet for R ceases to be a back facet for R'.
(2) A facet f which was not a back facet for R becomes a back facet for R'.
(3) The extreme vertex v changes to v'.

Note that in cases 1 and 2, the extreme vertex remains unchanged while in case 3 the set of back facets remains unchanged. This follows from the way we constructed $\mathcal{A}$. Also note that cases 1, 2 and 3 may occur simultaneously if $\mathcal{P}$ has parallel facets. In this case, we handle them one after the other.

In case 1, we obtain $V(R')$ from $V(R)$ in time $O(|f|)$, as follows: We first update the term $\frac{1}{2}(\mathbf{d} \cdot \mathbf{\Delta}(R))(\mathbf{d} \cdot \mathbf{v})$ by subtracting $\frac{1}{2}(\mathbf{d} \cdot \mathbf{\Delta}_f)(\mathbf{d} \cdot \mathbf{v})$. This can be done in $O(1)$ time since we have precomputed $\mathbf{\Delta}_f$. Then, in $O(|f|)$ time, we subtract $V'_f(R)$ from $V(R)$. Case 2 is handled similarly.

In case 3, we update $V(R)$ to $V(R')$ in $O(1)$ time by subtracting $\frac{1}{2}(\mathbf{d}\cdot\mathbf{\Delta}(R))(\mathbf{d}\cdot\mathbf{v})$ and adding $\frac{1}{2}(\mathbf{d} \cdot \mathbf{\Delta}(R'))(\mathbf{d} \cdot \mathbf{v}')$. (Note that, in this case, $\mathbf{\Delta}(R') = \mathbf{\Delta}(R)$ and is already known.) We also set v' to be the extreme vertex of R'.

How many times can a facet f appear or disappear as a back facet? This will happen each time that we cross, in the depth-first search, any edge of $\mathcal{A}$ which is contained in the great circle $h_f \cap \mathbb{S}^2$. Since there are $O(n)$ such edges and each is crossed only once, we conclude that f appears or disappears as a back facet $O(n)$ times. Therefore the total update time in case 1 and 2 is $O(\sum_{f \in \mathcal{P}} n|f|) = O(n^2)$. The total time spent in case 3 is clearly $O(n^2)$. Also, the total time spent in updating the constraints over all the faces of $\mathcal{A}$ is $O(n^2)$. We may now conclude:

Theorem 9 *Let $\mathcal{P}$ be a convex polyhedron of n vertices. A build direction which minimizes the total volume of supports needed by $\mathcal{P}$ can be computed in time $O(n^2)$.* ∎

4 Minimum-contact-area supports for a convex polyhedron

A facet $f \in \mathcal{P}$ needs support w.r.t. a direction $\mathbf{d}$ iff it is a back facet. Thus, the set of directions for which f needs support can be represented on the unit-sphere, $\mathbb{S}^2$, by an open hemisphere, h_f, whose pole is the point $-\mathbf{n}_f$ on $\mathbb{S}^2$.

We associate with h_f a weight equal to the area of f. Clearly, our problem is now equivalent to finding a point on $\mathbb{S}^2$ which is covered by hemispheres, h_f, of minimum total weight. We proceed as follows:

W.l.o.g. assume that the bounding plane of no hemisphere h_f is parallel to the xy-plane; this can be enforced in $O(n)$ time by rotating $\mathcal{P}$ about the x- or y-axis. Let $\mathbb{S}^2_+$ be the portion of $\mathbb{S}^2$ lying above the xy-plane and let $h^+_f = h_f \cap \mathbb{S}^2_+$. Using central projection [23], we map h^+_f to an open halfplane, ℓ^+_f, of the same weight, on the plane $z = 1$. Under this map, there is a 1–1 correspondence between the faces in the arrangement of the h^+_f's and the faces in the arrangement of the ℓ^+_f's. Thus, we can now solve our problem by finding a face in the latter arrangement which is covered by halfplanes of minimum total weight.

In [11, pages 181–182], Edelsbrunner and Guibas consider a similar problem: Given r doublewedges in the plane, find a point which is covered by the maximum number of doublewedges. Using topological sweep, they solve this problem in $O(r^2)$ time and $O(r)$ space. We can use this approach, with a minor modification to handle the halfplane weights, to solve our problem in $O(n^2)$ time and $O(n)$ space (since $r = O(n)$ in our case). We handle the portions of the hemispheres lying below the xy-plane in a symmetric way.

Theorem 10 *Let $\mathcal{P}$ be an n-vertex convex polyhedron. A direction $\mathbf{d}$ which*

minimizes the total surface area of $\mathcal{P}$ in contact with supports can be found in time $O(n^2)$, using $O(n)$ space. ∎

4.1 A faster algorithm when building $\mathcal{P}$ on a facet

To build $\mathcal{P}$ on a facet f, we must choose $-\mathbf{n}_f$ as the build direction. Let h_f be the closed hemisphere on $\mathbb{S}^2$ whose pole is the point $-\mathbf{n}_f$. Then, a facet $f' \neq f$ will need support iff $\mathbf{n}_{f'}$ is not contained in h_f. Let $C = \{\mathbf{n}_f | f \text{is a facet of } \mathcal{P}\}$ and let $C' = \{-\mathbf{n}_f | f \text{is a facet of } \mathcal{P}\}$. Associate with each point $\mathbf{n}_f \in C$ a weight equal to f's area. Thus, our problem is to find a point $-\mathbf{n}_f \in C'$ such that the total weight of the points $\mathbf{n}_{f'} \in C$ that are not in h_f is minimized, or, equivalently, the total weight of the points $\mathbf{n}_{f'} \in C$ that are in h_f is maximized.

It is convenient to reformulate our problem as follows: We are given $r = O(n)$ weighted *blue points* and r *red points* on $\mathbb{S}^2$, corresponding to the points in C and C', respectively. Each red point is the pole of a closed *red hemisphere*. We wish to find the red hemisphere which contains blue points of maximum total weight.

Let $P = (p_1, p_2, p_3)$ be any blue point. Assume w.l.o.g. $p_2 \neq 0$ for any blue point; this can be enforced in $O(n)$ time by rotating $\mathbb{S}^2$ suitably, about the first or third axis. Let $H : z = ax + by$ be the bounding plane of a red hemisphere; H passes through the origin. We map H to the red point $H' = (a, b)$ in the plane and map P to the blue line $P' : y = (-p_1/p_2)x + (p_3/p_2)$ in the plane.

Lemma 11 *Let P, H, H', and P' be as above.*

(A): P is on or above H iff ($p_2 > 0$ and H' is on or below P') or ($p_2 < 0$ and H' is on or above P').

(B): P is on or below H iff ($p_2 > 0$ and H' is on or above P') or ($p_2 < 0$ and H' is on or below P').

Proof.

(A): P is on or above H iff $p_3 \geq ap_1 + bp_2$; i.e., iff ($p_2 > 0$ and $b \leq (p_3/p_2) - a(p_1/p_2)$) or ($p_2 < 0$ and $b \geq (p_3/p_2) - a(p_1/p_2)$); i.e., iff ($p_2 > 0$ and H' is on or below P') or ($p_2 < 0$ and H' is on or above P').

The proof for (B) is similar. ∎

Let us divide the red hemispheres into two sets: those that have their poles above the equator of $\mathbb{S}^2$ (called *upper red hemispheres*) and those that have

their poles below (*lower red hemispheres*). Consider the upper red hemispheres.

The blue points P that are contained in any upper red hemisphere all lie on or above the plane H for that hemisphere. Therefore, by Lemma 11(A), our goal is to find a red point H' such that the total weight of the blue lines P' with $p_2 > 0$ that are on or above H' plus the total weight of the blue lines P' with $p_2 < 0$ that are on or below H' is maximum. (Similarly, Lemma 11(B) applies to the lower red hemispheres.)

We will use the following data structure, due to Matoušek, to solve the above problem. (See also Remark 15.)

Theorem 12 (Matoušek, [20]) *Let V be a set of n weighted points in $\mathbb{R}^d$, let m be a parameter such that $n \leq m \leq n^d$, let h be an integer such that $1 \leq h \leq d+1$, and let $\delta > 0$ be any real constant. In $O(n^{1+\delta} + m(\log n)^\delta)$ time, we can preprocess the points of V into a data structure of size $O(m)$ such that the total weight of the points of V lying in the intersection of any h halfspaces can be computed in time $O((n/m^{1/d})(\log \frac{m}{n})^{h-(d-h+1)/d})$.* ∎

Since the above data structure is designed for halfspace range counting, we need to dualize our problem once again. By a suitable duality transform (see, for instance, [10]), we can write Lemma 11 as follows, where H'' is the red line dual to point H' and P'' is the blue point dual to line P':

Lemma 13 *(A): P is on or above H iff ($p_2 > 0$ and P'' is on or below H'') or ($p_2 < 0$ and P'' is on or above H'').*

(B): P is on or below H iff ($p_2 > 0$ and P'' is on or above H'') or ($p_2 < 0$ and P'' is on or below H''). ∎

Note that Lemma 13(A) (resp. Lemma 13(B)) applies to the upper (resp. lower) red hemispheres.

We now build two data structures D^+ and D^-: D^+ is the structure of Theorem 12 built for $d = 2$ and $h = 1$ on the blue points P'' that correspond to points P with $p_2 > 0$. D^- is a similar structure for the blue points P'' with $p_2 < 0$. Thus, these structures can be used for halfplane queries on weighted points. For each upper red hemisphere with bounding plane H, we query D^+ with the lower halfplane of H'', query D^- with the upper halfplane of H'', and sum up the total weights returned. In this way, we find the upper red hemisphere that contains blue points of maximum total weight. We repeat this process with the lower red hemispheres also. This gives the overall optimal red hemisphere and thus yields an area-minimizing direction for building $\mathcal{P}$ on a facet.

Theorem 14 *Let $\mathcal{P}$ be an n-vertex convex polyhedron that is to be built on a facet. A build direction minimizing the total area of $\mathcal{P}$ that is in contact with supports can be found in $O(n^{4/3}(\log n)^{\gamma})$ time and space $O(n^{4/3}/(\log n)^{2\gamma})$, where $\gamma > 0$ is an arbitrarily small constant.*

Proof. Correctness is clear from the discussion above. For the running time, note that the weight computations and the transformations take $O(n)$ time. For any m, $n \leq m \leq n^2$, the structure of Theorem 12 uses $O(m)$ space and can be built in time $O(n^{1+\delta} + m(\log n)^{\delta})$. Each of the $r = O(n)$ queries takes time $O((n/m^{1/2})(\log \frac{m}{n})^{1-(2-1+1)/2}) = O(n/m^{1/2})$. Hence, the total time is $O(n^{1+\delta} + m(\log n)^{\delta} + n^2/m^{1/2})$, which is $O(n^{4/3}(\log n)^{\delta/3})$, if we choose $m = n^{4/3}/(\log n)^{2\delta/3}$. Setting $\gamma = \delta/3$ completes the proof. ∎

Remark 15 The data structure for halfplane range counting that we use in the above algorithm is quite intricate and so our algorithm is probably not very practical. We are not aware of any simple halfplane counting data structure that would yield an efficient algorithm for our problem. However, we can use the following practical data structure, due to Arya and Mount [2], which does *approximate* halfplane range counting: Let S be a set of n weighted points in the plane. Let Q be any bounded query range with bounded complexity and let its diameter be D. Let ε be a positive constant. If Q is convex, then with $O(n \log n)$ preprocessing and $O(n)$ space, we can answer a range counting query with Q in $O(\log n + 1/\varepsilon)$ time, such that the error is only $D \cdot \varepsilon$ (i.e., points within $D \cdot \varepsilon$ of Q's boundary may be misclassified as being inside/outside of Q.) To use this result for a halfplane query H (for which $D = \infty$), we enclose the point-set in an axes-parallel rectangle R and query with the range $H' = H \cap R$, which is convex and has bounded complexity. As pointed out to us [22] the error now is $D' \cdot \varepsilon$, where D' is the diameter of H'. Note that D' can be as large as the diameter of the point-set, but the error can be kept small by choosing ε suitably.

If we use this data structure in our algorithm, the overall running time is $O(n \log n + n/\varepsilon)$.

5 Conclusion

We have given efficient geometric algorithms for certain optimization problems arising in layered manufacturing. These include (a) simple and practical algorithms for minimizing the stair-step error for any polyhedron under different formulations of stair-step error, (b) an algorithm for minimizing the volume of the support structures for a convex polyhedron, and (c) an algorithm for minimizing the contact-area of supports for a convex polyhedron.

We believe that our quadratic algorithms for support optimization may not be improvable. We can show that a related problem belongs to the class of *3SUM-hard problems* [13] for which no subquadratic algorithms have been found despite much effort. Specifically, we are able to show that, for a convex polyhedron, it is 3SUM-hard to find a build direction $\mathbf{d}$ with a positive z-component which minimizes the number of facets needing support.

An interesting open problem that we are pursuing is the design of efficient algorithms for optimizing supports for a non-convex polyhedron. The fact that some facets may actually be attached to other facets instead of to the platform makes this problem complex. (See Figure 3.) In a companion paper [18], we have been successful in solving this problem in 2D for non-convex polygons. The idea is to divide the circle of directions into regions such that, within a region, the combinatorial structure of the supports does not change. We sweep over these regions, updating the support structure incrementally as we cross regions, and computing the best direction within each region by solving an optimization problem. In principle, our 2D techniques appear to carry over to 3D, but it is not clear at this point how efficient or practical this approach would be. Since no results are known for this problem, even a slow (brute-force) solution that is practical would be of interest. Another possibility is to devise a provably good approximation algorithm that computes a build direction for which the support volume is within some small factor of the support volume for the optimal direction.

In [18] we have also been able to find a build direction for 2D non-convex polygons which minimizes the volume of liquid resin that gets trapped when the polygon is built in that direction (the so-called "trapped volume" problem). Again, the analogous problem in 3D is of interest.

Another interesting problem is to simultaneously optimize two or more criteria. For instance, among all build directions that minimize the stair-step error, find the one which realizes the minimum volume for the supports. Or, find a build direction that allows the stair-step error and the number of layers to simultaneously meet designer-specified thresholds. Recently, in [19], we have presented efficient algorithms for several such multi-criteria optimization problems. These algorithms employ the solutions to the single-criterion optimization problems that we have presented here as building blocks.

Acknowledgements

We thank Larry Roscoe, Don Holzwarth, Jon Holt, Jeff Kotula, and Tom Studanski of Stratasys, Inc. for useful discussions and for providing us with test parts. We also thank the referees for helpful feedback.

References

[1] S. Allen and D. Dutta. Determination and evaluation of support structures in layered manufacturing. *Journal of Design and Manufacturing*, 5:153–162, 1995.

[2] S. Arya and D. M. Mount. Approximate Range Searching. In *Proc. 11th Annuual ACM Symposium on Computational Geometry*, pages 172–181, 1995.

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

[4] M. Bablani and A. Bagchi. Quantification of errors in rapid prototyping processes and determination of preferred orientation of parts. In *Transactions of the 23rd North American Manufacturing Research Conference*, 1995.

[5] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hull. Technical Report GCG53, Geometry Center, Univ. of Minnesota, July 1993. (See also `http://www.geom.umn.edu/software/qhull/`).

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

[7] B. Chazelle et al. Application challenges to computational geometry. Technical Report 521, Princeton University, April 1996. (Also available at `http://www.cs.duke.edu/~jeffe/compgeom/taskforce.html`).

[8] B. Chazelle, L.J. Guibas, and D.T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[9] A. Dolenc and I. Mäkelä. Slicing procedures for layered manufacturing techniques. *Computer-Aided Design*, 26:119–126, 1994.

[10] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer–Verlag, New York, 1987.

[11] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38:165–194, 1989.

[12] D. Frank and G. Fadel. Preferred direction of build for rapid prototyping processes. In *Proceedings of the 5th International Conference on Rapid Prototyping*, pages 191–200, 1994.

[13] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory & Applications*, 5:165–185, 1995.

[14] R.N. Goldman. Area of planar polygons and volume of polyhedra. In J. Arvo, editor, *Graphics Gems II*, pages 170–171. Academic Press, 1991.

[15] P. Jacobs. *Rapid Prototyping & Manufacturing: Fundamentals of StereoLithography.* McGraw-Hill, 1992.

[16] D.G. Luenberger. *Introduction to linear and non-linear programming.* Addison-Wesley, 1973.

[17] J. Majhi, R. Janardan, M. Smid and P. Gupta. On some geometric optimization problems in layered manufacturing. In *Proceedings of the 5th Workshop on Algorithms and Data Structures*, volume 1272, pages 136–149, Springer-Verlag, 1997.

[18] 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*, to appear.

[19] J. Majhi, R. Janardan, M. Smid and J. Schwerdt. Multi-criteria geometric optimization problems in Layered Manufacturing. In *Proceedings of the 14th Annual ACM Symposum on Computational Geometry*, pages 19-28, 1998.

[20] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10:157–182, 1992.

[21] M. McKenna and R. Seidel. Finding the optimal shadows of a convex polytope. In *Proceedings of the 1st Annual ACM Symposium on Computational Geometry*, pages 24–28, 1985.

[22] D. M. Mount. Private communication, Dec 1997.

[23] F.P. Preparata and M.I. Shamos. *Computational Geometry–An Introduction.* Springer–Verlag, 1988.