ELSEVIER

Computational Geometry 13 (1999) 229–252

Computational Geometry
Theory and Applications

www. elsevier.nl/locate/comgeo

Computing a flattest, undercut-free parting line for a convex polyhedron, with application to mold design ☆

Jayanth Majhi [a,1], Prosenjit Gupta [b,2], Ravi Janardan [c,*]

[a] *Mentor Graphics Corp., 8005 S.W. Boeckman Rd., Wilsonville, OR 97070, USA*

[b] *Delsoft (India), Pvt. Ltd., India*

[c] *Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA*

Communicated by G.T. Toussaint; submitted 28 April 1997; accepted 10 August 1999

Abstract

A parting line for a polyhedron is a closed curve on its surface, which identifies the two halves of the polyhedron for which mold-boxes must be made. A parting line is undercut-free if the two halves that it generates do not contain facets that obstruct the de-molding of the polyhedron. Computing an undercut-free parting line that is as "flat" as possible is an important problem in mold design. In this paper, algorithms are presented to compute such a parting line for a convex polyhedron, based on different flatness criteria. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Casting/molding; Computational geometry; Optimization; Arrangements; Shortest paths; Visibility; Point-set width

1. Introduction

We consider a geometric problem arising in the design of molds for casting and injection molding. Consider the construction of a sand mold for casting a polyhedral solid. First a prototype, $\mathcal{P}$, of the polyhedron is made. Two halves of $\mathcal{P}$ are then identified and a separate mold-box is made for each. This involves placing $\mathcal{P}$ in a box and packing sand around the first half. After the sand has been compacted and hardened, $\mathcal{P}$ is translated out of the mold-box, i.e., *de-molded*, and a second mold-box is made similarly for the other half of $\mathcal{P}$. The two mold-boxes are then fastened together by pins to form a cavity in the

☆ Research supported in part by NSF Grant CCR-9200270 and by a University of Minnesota Grant-in-Aid of Research Award. A preliminary version of this paper appears in the Proceedings of the First ACM Workshop on Applied Computational Geometry, LNCS 1148, Springer-Verlag, 1996, pp. 109–120.

* Corresponding author. E-mail: janardan@cs.umn.edu

[1] E-mail: jayanth-majhi@mentorg.com

[2] Work done while at MPI-Informatik, Saarbrücken, Germany, and at the University of Minnesota, USA. E-mail: pjit@excite.com

PII: S0925-7721(99)00023-1

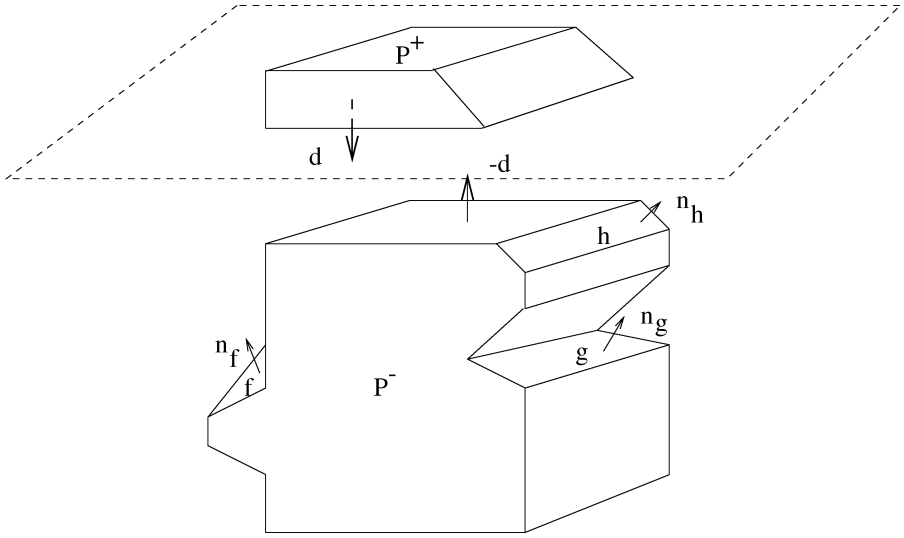

Fig. 1. Illustrating undercuts.

shape of $\mathcal{P}$ and molten metal is poured into it. (More details can be found in [21].)

Among the many key issues surrounding the design of a good mold, two that are mentioned extensively in the literature are (i) the shape of the parting line, and (ii) the number of undercuts.

The *parting line* is a continuous closed curve on the surface of $\mathcal{P}$ which defines the two halves; thus it also defines the line of contact between the two mold-boxes. As noted in [2,21], the parting line should be chosen to be as "flat" as possible since this results in a more cost-efficient and accurate mold.[3] A highly stepped parting line calls for very skilled mold makers, can cause instability of the mold, and can result in seepage of molten material at the line of contact—all of which lead to higher production costs.

An *undercut* is a facet (or a portion thereof) of $\mathcal{P}$ whose outward normal makes an angle less than 90° with the de-molding direction for the half of $\mathcal{P}$ containing the facet. Undercuts cause obstructions when the polyhedron is de-molded and are hence undesirable.

Example. Fig. 1 depicts a polyhedron, $\mathcal{P}$, divided into two halves, $\mathcal{P}^+$ and $\mathcal{P}^-$ by a plane (shown dashed). P^+ is de-molded in direction $\boldsymbol{d}$ and $\mathcal{P}^-$ in direction $-\boldsymbol{d}$. Note that the parting line here is a rectangle and it lies in a plane. For this choice of de-molding direction and parting line, there are three undercuts in $\mathcal{P}^-$, namely f, g, and h (which is part of a larger facet), since their respective normals, $\boldsymbol{n}_f, \boldsymbol{n}_g$, and $\boldsymbol{n}_h$, make angles less than 90° with $-\boldsymbol{d}$. Since the mold-box for $\mathcal{P}^-$ will be filled with sand up to the parting line, these facets will cause obstructions when de-molding $\mathcal{P}^-$. There are no undercuts in P^+.

Notice that if we had picked the parting plane to coincide with the lower edge of h, then h would belong to the upper half and would no longer be an undercut. Notice also that if we had parted $\mathcal{P}$ down the middle with a vertical plane that is normal to the paper and de-molded the halves to the left and to the right, then there would be no undercuts at all.

[3] Intuitively, by a "flat" parting line, we mean one which lies as nearly as possible in a plane. We will formalize this notion in Section 2.

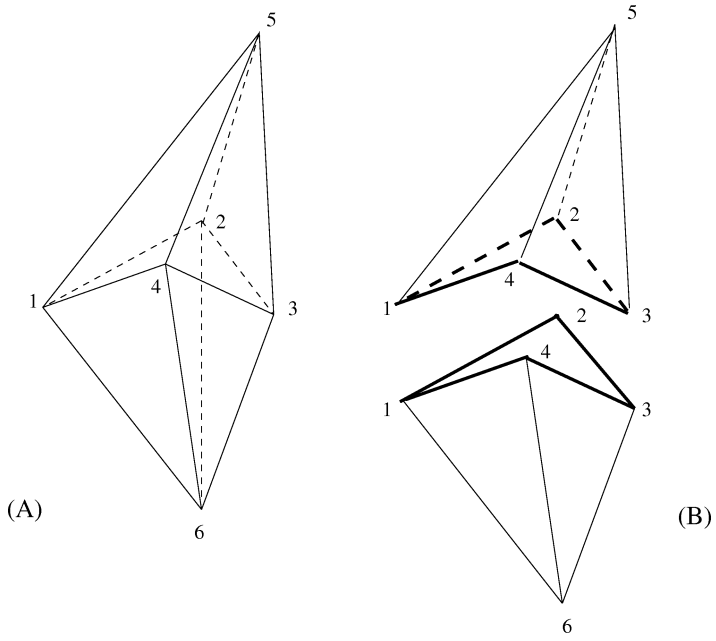

Fig. 2. (a) A convex polyhedron which admits no undercut-free parting line lying in a plane. (b) A nonplanar undercut-free parting line, shown in heavy lines.

The adverse effects of undercuts can be alleviated by using so-called cores and inserts in the mold. However, this is generally discouraged since it increases the cost of the mold and slows down part production [7,21]. Thus it is important to minimize the number of undercuts.

A judicious choice of the de-molding direction and the parting line can often reduce or altogether eliminate undercuts, as the simple example above illustrates. However, in general, the reduction in undercuts is usually accompanied by an increase in the complexity of the parting line. In fact, this is true even if $\mathcal{P}$ is convex, as the following example shows.

Example. Fig. 2(a) shows an octahedron $\mathcal{P}$. It is not difficult to see that any plane which divides it into two creates undercuts. For instance, the plane determined by vertices 1, 2, and 3 creates undercuts for the upper half when de-molding this half downwards; these undercuts are the portions of facets $(1, 4, 6)$ and $(3, 4, 6)$ that belong to the upper half. As shown in Fig. 2(b), undercuts can be avoided altogether by choosing the parting line to be the chain 1–2–3–4–1 (or 2–5–4–6–2); however, the parting line is no longer flat.

Intuitively, it should be clear that a convex polyhedron always has an undercut-free parting line for any choice of de-molding directions $\boldsymbol{d}$ and $-\boldsymbol{d}$. In particular, the boundary of $\mathcal{P}$ when it is viewed along lines of sight parallel to $\boldsymbol{d}$ is such a line. (We will make this more precise in Section 2.1.) However, different directions can yield different parting lines. In this paper, we consider the problem of finding the flattest, undercut-free parting line for a convex polyhedron, $\mathcal{P}$, based on two types of flatness criteria that we discuss in Section 2.2. The first type takes into account the length of the parting line in relation to the

length of its projection, while the second considers the relative displacement of the vertices of the parting line (i.e., the "width" of the parting line).

1.1. Related work

We are not aware of prior work on the specific problem that we consider; however, we mention briefly some related work: Bose et al. [3] (see also [4]) give efficient algorithms to decide if a given polyhedron admits an undercut-free parting line which lies in a plane. Chen et al. [7] consider the problem of finding a de-molding direction for a polyhedron which minimizes the number of undercuts. In [1], Ahn et al. give algorithms for deciding if the mold for a polyhedron can be decomposed into two parts such that the polyhedron can be de-molded in a given direction without getting stuck. They also give algorithms for computing all de-molding directions for which this is possible. Similar problems are addressed in 2D by Rosenbloom and Rappaport [18]. However, the results in [1,7] do not consider the shape of the parting line, which can be quite stepped. In [13], Hui and Tan give heuristics for finding parting directions with few (but not necessarily minimum) undercuts. Ravi and Srinivasan [17] identify several criteria (different from ours) for the design of good parting lines but do not give any algorithms for computing these lines.

We close by mentioning some other geometric work of interest in the area of mold design. In [4–6], the general problem of "mold fillability" is addressed. The questions of interest here include deciding whether a given mold can be filled from a given "pour" direction without creating air pockets, determining all such pour directions, computing a direction that minimizes the number of air pockets (if air pockets are unavoidable), and characterizing classes of polyhedra with respect to their fillability. Efficient algorithms are given in [4,5] for 2-dimensional molds and in [4,6] for 3-dimensional molds. In [10], related questions are also addressed for different mold-filling strategies and different filling materials.

1.2. Organization of the paper

In Section 2 we make precise the notions of parting line and flatness. In Section 3 we describe our solution for the length-based flatness criterion and in Section 4 we discuss our implementation of this solution. In Section 5, we discuss briefly a solution to a variant of the length-based criterion. In Section 6 we discuss our solution for the width-based criterion. We conclude in Section 7.

2. Parting line and flatness criteria

2.1. Parting line

Let $\mathcal{P}$ be our convex polyhedron and let $\boldsymbol{d}$ be any direction. Suppose that we view $\mathcal{P}$ from infinity along lines of sight that are parallel to $\boldsymbol{d}$. A point p on $\mathcal{P}$ is *$\boldsymbol{d}$-visible* if the ray from p in direction $-\boldsymbol{d}$ misses the interior of $\mathcal{P}$. A facet f of $\mathcal{P}$ is *$\boldsymbol{d}$-visible* if every point of f is $\boldsymbol{d}$-visible. Since $\mathcal{P}$ is convex, this is equivalent to saying that the angle between $-\boldsymbol{d}$ and the outward normal, $\boldsymbol{n}_f$, to f is at most 90°, i.e., $(-\boldsymbol{d})\cdot\boldsymbol{n}_f \geqslant 0$. Note that a $\boldsymbol{d}$-visible facet f will not create an undercut when the half of $\mathcal{P}$ containing it is de-molded along direction $\boldsymbol{d}$ (since the angle between $\boldsymbol{n}_f$ and $\boldsymbol{d}$ is at least 90°).

Let $F(\boldsymbol{d})$ be the set of $\boldsymbol{d}$-visible facets and let $B(\boldsymbol{d})$ be the boundary of their union. $B(\boldsymbol{d})$ is a closed chain consisting of the edges of $\mathcal{P}$. We take $B(\boldsymbol{d})$ to be the *parting line* for $\mathcal{P}$, with respect to de-molding

$\mathcal{P}$ along directions $\boldsymbol{d}$ and $-\boldsymbol{d}$, and we denote it by $L(\boldsymbol{d})$. Since the facets on one side of $L(\boldsymbol{d})$ are all $\boldsymbol{d}$-visible and the facets on the other side are all $(-\boldsymbol{d})$-visible, it follows that $L(\boldsymbol{d})$ is free of undercuts with respect to de-molding these two halves in directions $\boldsymbol{d}$ and $-\boldsymbol{d}$, respectively.

Example. Consider the polyhedron in Fig. 2. Let $\boldsymbol{d}$ be vertically downwards. Then $F(\boldsymbol{d})$ consists of the facets $(1,2,5)$, $(2,3,5)$, $(3,4,5)$, and $(1,4,5)$, and $B(\boldsymbol{d}) = L(\boldsymbol{d})$ consists of the line segments $(1,2)$, $(2,3)$, $(3,4)$, and $(4,1)$.

2.2. Flatness criteria

Criterion ρ. Let $L(\boldsymbol{d}) = e_1, e_2, \ldots, e_k$, where the e_i's are line segments. Let $\widehat{L}(\boldsymbol{d}) = \widehat{e}_1, \widehat{e}_2, \ldots, \widehat{e}_k$ be the parallel projection of $L(\boldsymbol{d})$ onto a plane normal to $\boldsymbol{d}$. Note that while $\widehat{L}(\boldsymbol{d})$ lies in a plane, $L(\boldsymbol{d})$ can be highly stepped since the e_i's can zigzag considerably in 3-space. We measure the flatness of $L(\boldsymbol{d})$ in direction $\boldsymbol{d}$ by

$$\rho(\boldsymbol{d}) = \sum_{i=1}^{k} \text{length}(\widehat{e}_i)^2 \Big/ \sum_{i=1}^{k} \text{length}(e_i)^2, \tag{1}$$

where length(l) is the Euclidean length of segment l. Note that $\rho(d) \leqslant 1$, with equality holding iff $L(\boldsymbol{d})$ lies in a plane. In general, the larger the value of $\rho(\boldsymbol{d})$, the flatter is $L(\boldsymbol{d})$. *Our goal is to find a $\boldsymbol{d}$ which maximizes $\rho(\boldsymbol{d})$.*

Criterion ω. Define the *width* of $L(\boldsymbol{d})$ in direction $\boldsymbol{d}$—denoted by $\omega(\boldsymbol{d})$—as the smallest distance between two parallel planes that are normal to $\boldsymbol{d}$ and enclose $L(\boldsymbol{d})$. We measure the flatness of $L(\boldsymbol{d})$ in direction $\boldsymbol{d}$ by $\omega(d)$. Clearly, $L(\boldsymbol{d})$ lies in a plane iff $\omega(\boldsymbol{d}) = 0$. In general, the smaller the value of $\omega(\boldsymbol{d})$, the flatter is $L(\boldsymbol{d})$. *Our goal is to find a $\boldsymbol{d}$ which minimizes $\omega(\boldsymbol{d})$.*

Remark 2.1. We remark that our width problem is a constrained version of the conventional width problem [12], in the following sense. In the conventional problem, the structure whose width we wish to compute remains fixed as we search over the space of all directions for the best pair of enclosing planes. As we will see, in our case the parting line L, whose width we wish to compute, remains fixed only over a certain "region" of the space of directions. Therefore, for each L, we need to restrict our search for the best pair of enclosing planes to the corresponding region.

Discussion. Criteria ρ and ω are representative of two classes of flatness measures—one based on the lengths of the segments comprising the parting line and the other on the relative positions of its vertices. Both give an indication of how close a parting line is to lying in a plane. Our algorithm is sufficiently general in that it allows us to "plug in" other flatness measures quite easily. For instance, one variant on Criterion ρ is *Criterion ρ'*, defined as

$$\rho'(\boldsymbol{d}) = \sum_{i=1}^{k} \text{length}(\widehat{e}_i) \Big/ \sum_{i=1}^{k} \text{length}(e_i). \tag{2}$$

That is, we use the sum of edge lengths rather than the sum of their squares. Indeed, we have implemented a version of our algorithm for both ρ and ρ' and switching from one to the other required very little change in the code.

Computationally, however, Criterion ρ' is much more difficult to work with because it gives rise to highly nonlinear optimization problems (owing to the use of square roots in computing $\mathrm{len}(\cdot)$) and because it does not facilitate easy updating of the parting line as we search over the space of directions. (We will elaborate on this in Section 5.) On the other hand, the corresponding problems for Criterion ρ exhibit nice structure that allow us to bring into play interesting algorithmic techniques that solve the problem efficiently (Sections 3.2, 3.3 and 3.6). Therefore, we have chosen to describe in detail our algorithm for the length-based criterion in the context of ρ rather than ρ'.

We close this section with a formal statement of the problem that we wish to solve.

Problem 1. Given a convex polyhedron $\mathcal{P}$ with n vertices, find a direction $\boldsymbol{d}$ such that
(i) $\rho(\boldsymbol{d})$ is maximized, or
(ii) $\omega(\boldsymbol{d})$ is minimized.

2.3. Parting lines revisited

There are situations where several undercut-free parting lines exist, in addition to the one given by the definition in Section 2.1. In such cases, it is advantageous to choose among these lines, the one which most favors the flatness criterion being used.

Let $\boldsymbol{d}$ be any direction and call a facet $f \in \mathcal{P}$ $\boldsymbol{d}$*-parallel* if it is parallel to $\boldsymbol{d}$. Suppose $\mathcal{P}$ has one or more $\boldsymbol{d}$-parallel facets, f. By definition, $f \in F(\boldsymbol{d})$. Now $B(\boldsymbol{d})$ is of the form $c_1F_1c_2F_2\ldots c_mF_m$, for some m. Here each c_i is a chain of edges of $\mathcal{P}$, with endpoints u_i and v_i and each F_i is a group of contiguous $\boldsymbol{d}$-parallel facets, attached to c_i at v_i and to c_{i+1} at u_{i+1}—indices are taken modulo m.

Fig. 3 illustrates the situation. Notice that there are several ways of choosing an undercut-free parting line, depending on how we join the v_i's and the u_{i+1}'s. Our goal is to find a path of line segments from v_i to u_{i+1}, one segment per facet of F_i, such that $\rho(\boldsymbol{d})$ is maximized.

Observe that $\widehat{L}(\boldsymbol{d})$ is the same regardless of how we join v_i and u_{i+1}, because it is a projection in direction $\boldsymbol{d}$. Therefore, we must choose $L(\boldsymbol{d})$ such that we minimize the denominator in Eq. (1). This can be accomplished by picking the path from v_i to u_{i+1} such that the sum of the squares the segment

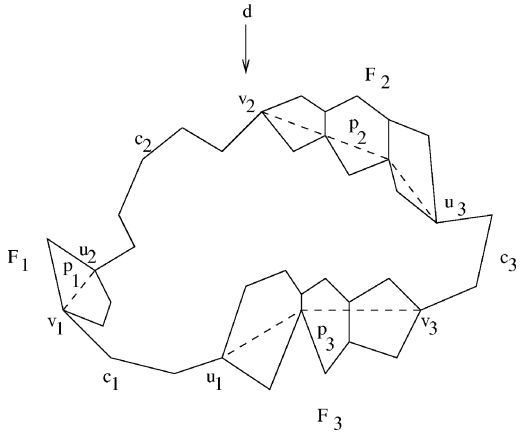

Fig. 3. The parting line in the presence of $\boldsymbol{d}$-parallel facets.

lengths on this path is minimum. However, it is not clear how such a path can be computed efficiently using standard geometric shortest path algorithms, since the "sum-of-squares" is not a metric (it violates the triangle inequality). Instead, we pick the path from v_i to u_{i+1} to be an Euclidean shortest path, which we denote by p_i. Thus, $L(\boldsymbol{d}) = e_1, e_2, \ldots, e_k$, where each e_j is either an edge of a c_i or a line segment of p_i. We will see in Section 3.4 how to compute p_i efficiently. It should be borne in mind that when we speak subsequently of maximizing Criterion ρ (Theorems 3.1 and 3.2), it is in the context of this choice of $L(\boldsymbol{d})$. (We opted to use the shortest path since this is needed anyway for the analogous situation which arises with Criterion ρ'. In that case, of course, the shortest path does indeed minimize the denominator in Eq. (2).)

For Criterion ω, we join each v_i to u_{i+1} in such a way that the width of the resulting closed chain is minimum. We will show later, in Section 6.3, how such a path can be computed efficiently.

2.4. Overview of the result

We give an algorithm for Problem 1(i) which runs in $O(n^2)$ time. The algorithm employs a combination of continuous and discrete optimization. Briefly, our approach is as follows. We first subdivide 3-space into $O(n^2)$ unbounded, interior-disjoint polyhedral regions (called *cones*), each apexed at the origin. Each cone has the property that $L(\boldsymbol{d})$ is the same for all directions $\boldsymbol{d}$ in the interior of the cone. Thus, maximizing $\rho(\boldsymbol{d})$ inside a cone is equivalent to maximizing the numerator in Eq. (1), which gives rise to a continuous optimization problem for each cone, as discussed in Section 3.2.

Similarly, $L(\boldsymbol{d})$ is the same for all directions $\boldsymbol{d}$ that lie on a bounding plane of a cone. However, now $\mathcal{P}$ will have $\boldsymbol{d}$-parallel facets, so that we also need to perform certain shortest path computations on the surface of $\mathcal{P}$ to compute $L(\boldsymbol{d})$. It turns out that this problem can be formulated as a shortest path problem on a special planar polygon and hence can be solved quickly.

Thus, the idea is to compute the ρ-value for each cone and cone boundary and pick the best one. However, a direct implementation of this method is not efficient because formulating an optimization problem for a cone or cone boundary requires knowledge of the parting line, and computing the latter from scratch each time takes $O(n)$ time per parting line and results in an $O(n^3)$ algorithm. We circumvent this problem by visiting the cones in a certain order and updating the parting line incrementally, so that the total time reduces to $O(n^2)$.

For Problem 1(ii), we give an $O(n^4)$-time algorithm. Once again, we divide 3-space into cones such that the parting line is the same for all directions in the interior of a cone or in the interior of a cone boundary. Within each such region we show how to solve a constrained version of the conventional width problem [12], in $O(n^2)$ time, and pick the direction which yields the smallest overall value for ω.

3. Flattest undercut-free parting line under Criterion ρ

3.1. Subdividing 3-space into cones

We follow the approach in [15]: For each facet f of $\mathcal{P}$, we construct a plane, h_f, which is parallel to f and passes through the origin. The planes h_f subdivide 3-space into the afore-mentioned collection of cones. Fig. 4 illustrates a cone. The following lemma establishes an upper bound on the number of cones.

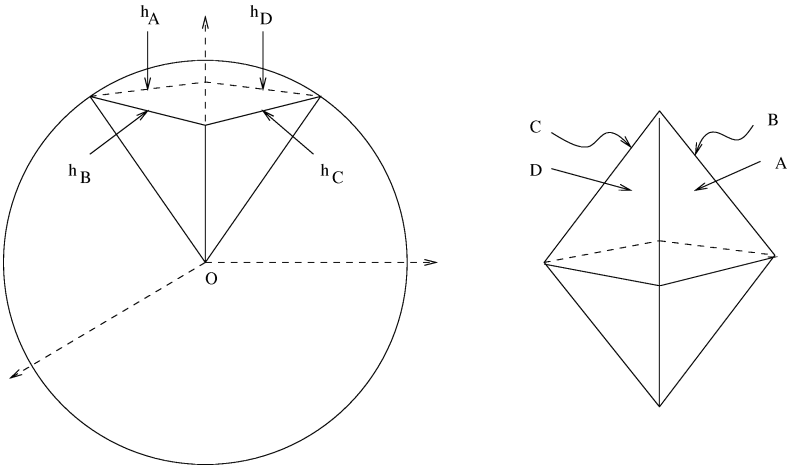

Fig. 4. A cone in the decomposition of 3-space for an octahedron.

Lemma 3.1. *There are* $O(n^2)$ *cones.*

Proof. From Euler's relation for convex polyhedra [8], it follows that $\mathcal{P}$ has $O(n)$ facets. Hence there are $O(n)$ planes h_f.

Consider the unit-sphere, $\mathbb{S}^2$, centered at the origin. Let g_f be the great circle $h_f \cap \mathbb{S}^2$, for each facet f of $\mathcal{P}$. Since two distinct great circles intersect exactly twice, the arrangement of the g_f's yields a planar subdivision, $\mathcal{A}$, of $\mathbb{S}^2$, with $O(n^2)$ vertices, and, hence, $O(n^2)$ faces and edges.

Let C be any cone. Since C is unbounded, it intersects $\mathbb{S}^2$ and hence corresponds to the unique face $C \cap \mathbb{S}^2$ of $\mathcal{A}$. Let r be any face of $\mathcal{A}$. Then r corresponds to at most one cone, since the interiors of cones are pairwise disjoint. Moreover, if r is bounded by great circles $g_{f_1}, g_{f_2}, \ldots$, then it corresponds to at least one cone, namely the cone bounded by the planes $h_{f_1}, h_{f_2}, \ldots$. Hence, there is a unique cone corresponding to r.

The above argument establishes a bijection between the cones and the faces of $\mathcal{A}$ and the lemma follows. □

In view of the above bijection, we can work with the arrangement $\mathcal{A}$ on $\mathbb{S}^2$, rather than with unbounded cones. For convenience, we will hereafter use the term *cone* to mean a face of $\mathcal{A}$ and the terms *edge/vertex of a cone* to mean an edge/vertex of the face. Note that a direction $\boldsymbol{d}$ is now a point on $\mathbb{S}^2$ and is hence a unit-vector.

The following lemmas establish two crucial properties of the cones.

Lemma 3.2. *Let* C *be any cone.* $L(\boldsymbol{d})$ *is the same for all points* $\boldsymbol{d}$ *in the interior of* C.

Proof. Let $\boldsymbol{d}$ and $\boldsymbol{d}'$ be distinct points in the interior of C and let f be any facet of $\mathcal{P}$. Then $\boldsymbol{d}$ and $\boldsymbol{d}'$ are both on the same side of h_f. Thus, $(-\boldsymbol{d}) \cdot \boldsymbol{n}_f$ and $(-\boldsymbol{d}') \cdot \boldsymbol{n}_f$ are both positive or both negative, and so f is $\boldsymbol{d}$-visible iff f is $\boldsymbol{d}'$-visible. Therefore, $F(\boldsymbol{d}) = F(\boldsymbol{d}')$ and hence $B(\boldsymbol{d}) = B(\boldsymbol{d}')$. Since $\mathcal{P}$ does not have any $\boldsymbol{d}$-parallel or $\boldsymbol{d}'$-parallel facets, $B(\boldsymbol{d})$ and $B(\boldsymbol{d}')$ consist of edges of $\mathcal{P}$. Therefore, by the discussion in Section 2, we have $L(\boldsymbol{d}) = B(\boldsymbol{d})$ and $L(\boldsymbol{d}') = B(\boldsymbol{d}')$. Thus $L(\boldsymbol{d}) = L(\boldsymbol{d}')$ and the lemma follows. □

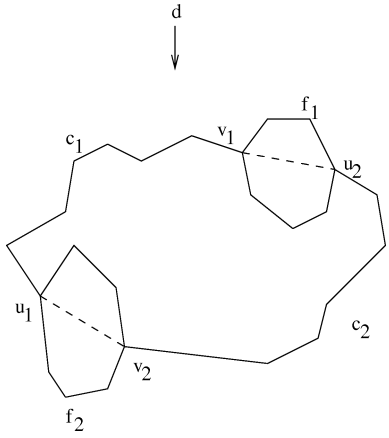

Fig. 5. Structure of $B(\boldsymbol{d})$ when $\boldsymbol{d}$ lies in the interior of an edge.

Lemma 3.3. *Let e be any edge of a cone C. $L(\boldsymbol{d})$ is the same for all points $\boldsymbol{d}$ in the interior of e.*

Proof. Since $\boldsymbol{d}$ and $\boldsymbol{d}'$ are in the interior of an edge, they belong to exactly one great circle g. $\mathcal{P}$ can have at most two facets f_1 and f_2 such that $h_{f_1} \cap \mathbb{S}^2 = h_{f_2} \cap \mathbb{S}^2 = g$. Thus, f_1 and f_2 are the only $\boldsymbol{d}$-parallel and the only $\boldsymbol{d}'$-parallel facets of $\mathcal{P}$. Moreover, because f_1 and f_2 are mutually parallel, they must be disjoint.

Proceeding along the lines of Lemma 3.2, we can show that $F(\boldsymbol{d}) = F(\boldsymbol{d}')$ and $B(\boldsymbol{d}) = B(\boldsymbol{d}')$. Let us now examine the structure of $B(\boldsymbol{d})$ and $B(\boldsymbol{d}')$.

Assume that both f_1 and f_2 exist (the case where only one exists is similar). By the discussion in Section 2, $B(\boldsymbol{d})$ consists of: (i) a chain c_1 of edges of $\mathcal{P}$ starting at a vertex u_1 of f_2 and ending at a vertex v_1 of f_1, (ii) a chain c_2 of edges of $\mathcal{P}$ starting at a vertex u_2 of f_1 and ending at a vertex v_2 of f_2, and (iii) the facets f_1 and f_2. (See Fig. 5.) Thus, $L(\boldsymbol{d})$ consists of c_1, c_2, and the straight-line segments $\overline{v_1u_2} \in f_1$ and $\overline{v_2u_1} \in f_2$, which constitute minimum-length paths between their respective endpoints.

Since $B(\boldsymbol{d}') = B(\boldsymbol{d})$, it follows that $L(\boldsymbol{d}')$ also consists of c_1, c_2, $\overline{v_1u_2}$, and $\overline{v_2u_1}$. The lemma follows. □

We denote the unique parting line associated with the interior of C (respectively interior of e) by $L(C)$ (respectively $L(e)$). Trivially, if v is a vertex of C, then the associated parting line is unique; we denote this by $L(v)$. It is clear now that to maximize $\rho(\boldsymbol{d})$ in the interior of C, we need only find a $\boldsymbol{d}$ in C's interior such that $\widehat{L}(\boldsymbol{d})$ is maximized. We formulate this optimization problem below. A similar approach is taken to maximize $\rho(\boldsymbol{d})$ in the interior of e.

3.2. The optimization problem

Let $\boldsymbol{d} \in \mathbb{S}^2$ and let C be any cone. Let $L(C) = \boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_k$, where each e_ℓ is an edge of $\mathcal{P}$ (with orientation assigned arbitrarily). Assume that $\boldsymbol{d} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$ and $\boldsymbol{e}_\ell = a_\ell\boldsymbol{i} + b_\ell\boldsymbol{j} + c_\ell\boldsymbol{k}$, where $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ are unit-vectors along the x-, y-, and z-axes. Let θ_ℓ be the angle between $\boldsymbol{e}_\ell$ and $\boldsymbol{d}$.

We have

$$\text{length}(\widehat{\boldsymbol{e}}_\ell)^2 = \big(\text{length}(\boldsymbol{e}_\ell) \cdot \sin\theta_\ell\big)^2 = \big(\text{length}(\boldsymbol{e}_\ell \times \boldsymbol{d})\big)^2.$$

Since

$$\boldsymbol{e}_\ell \times \boldsymbol{d} = (b_\ell z - c_\ell y)\boldsymbol{i} + (c_\ell x - a_\ell z)\boldsymbol{j} + (a_\ell y - b_\ell x)\boldsymbol{k},$$

we have

$$\text{length}(\widehat{\boldsymbol{e}}_\ell)^2 = A_\ell x^2 + B_\ell y^2 + C_\ell z^2 + D_\ell xy + E_\ell yz + F_\ell xz,$$

where

$$A_\ell = c_\ell^2 + b_\ell^2, \qquad B_\ell = a_\ell^2 + c_\ell^2, \qquad C_\ell = a_\ell^2 + b_\ell^2,$$
$$D_\ell = -2a_\ell b_\ell, \qquad E_\ell = -2b_\ell c_\ell, \qquad F_\ell = -2a_\ell c_\ell.$$

Thus,

$$\sum_{\ell=1}^{k} \text{length}(\widehat{\boldsymbol{e}}_\ell)^2 = Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fzx,$$

where $A = \sum_{\ell=1}^{k} \mathcal{A}_\ell$, and similarly for B through F.

Let $\boldsymbol{d}_C \in \mathbb{S}^2$ be a given point in C's interior. ($\boldsymbol{d}_C$ is computed at the time $\mathcal{A}$ is constructed.) Let $\boldsymbol{n}_f$ be the outward-directed normal to facet f. Note that $\boldsymbol{d}$ is in the interior of C iff $\boldsymbol{d} \cdot \boldsymbol{n}_f$ and $\boldsymbol{d}_C \cdot \boldsymbol{n}_f$ are both positive or both negative for each great circle g_f bounding C.

Thus our optimization problem for the interior of C is:

$$\begin{aligned}
&\text{maximize} && f(x, y, z) = Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz \\
&\text{subject to} && x^2 + y^2 + z^2 = 1 \quad \text{(sphere constraint)}, \\
& && \boldsymbol{d} \cdot \boldsymbol{n}_f > 0 \text{ (respectively } \boldsymbol{d} \cdot \boldsymbol{n}_f < 0) \text{ if } \boldsymbol{d}_C \cdot \boldsymbol{n}_f > 0 \text{ (respectively } \boldsymbol{d}_C \cdot \boldsymbol{n}_f < 0) \\
& && \quad \text{for each great circle } g_f \text{ bounding } C \quad \text{(plane constraints)}.
\end{aligned}$$

The optimization problem for the interior of an edge e is formulated similarly. However, there are just three plane constraints now—one requiring that the solution point lie on e's great circle and the other two requiring that it lie in the interior of e. For a vertex v, there is no need to solve any optimization problem, since it is a single point.

3.3. Solving the optimization problem

We use the method of Lagrange Multipliers [14]. In more detail, consider the optimization problem for cone C. 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, with respect to 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 later be computing the parting line at edge interiors and at vertices. If case (i) holds then we check if either of the two points lies in C (by checking the plane constraints) and, if so, then we compute the corresponding value of ρ in the interior of C.

For the interior of an edge, e, which lies on a great circle defined by the plane $ax + by + cz = 0$, 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 find the corresponding values of ρ.

Analysis. It is reasonable to assume that the cubic and quadratic equations that arise can be solved in O(1) time. Thus, the optimization problem for C (respectively e) takes time $O(|C|)$ (respectively O(1)). Summed over all cones and edges, this is $O(n^2)$.

Remark 3.1. If some other optimization algorithm is to be used, it would be advantageous to first convert our optimization problem to one of constant size, i.e., one where all of the following are constant:

(i) the number of variables,
(ii) the description size of the objective function and of each constraint, and
(iii) the number of constraints. It might then be reasonable to assume that each such problem can be solved in constant time, for a time bound of $O(n^2)$ for all problems.

Conditions (i) and (ii) already hold in our case. To enforce condition (iii), we can triangulate the cones into a total of $O(n^2)$ subcones, which is easy to do since the cones are convex. The number of optimization problems is still $O(n^2)$, but now each has just four constraints.

*3.4. Handling **d**-parallel facets*

As discussed in Section 2.3, in the presence of $\boldsymbol{d}$-parallel facets, portions of the parting line need to be computed as shortest paths. $\mathcal{P}$ has $\boldsymbol{d}$-parallel facets if $\boldsymbol{d}$ is in the interior of an edge $e \in \mathcal{A}$ or if $\boldsymbol{d}$ is a vertex of $\mathcal{A}$. In the former case, there are at most two groups, F_1 and F_2, of $\boldsymbol{d}$-parallel facets, and each F_i consists of just one facet f_i, $i = 1, 2$. Thus, p_i is just the line segment $\overline{v_i u_{i+1}}$. (See Fig. 5.)

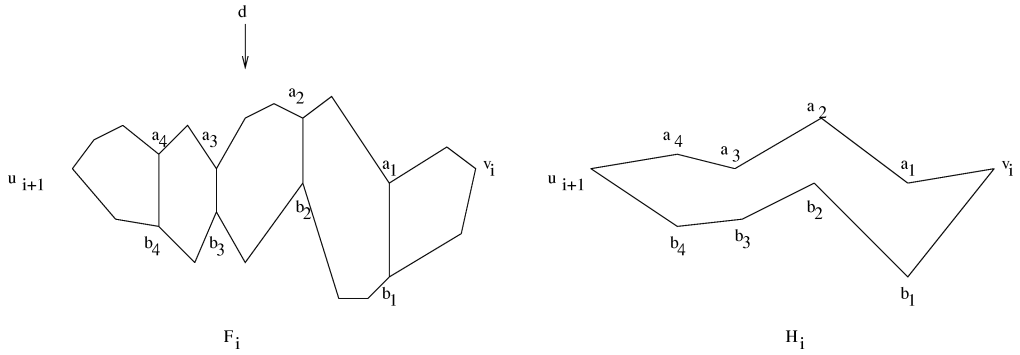

Fig. 6. Formulating the 2-dimensional shortest path problem.

In the latter case, there can be many groups F_i, each made up of several contiguous $\boldsymbol{d}$-parallel facets (see Fig. 3). The computation of p_i can now be formulated as a 2-dimensional shortest path problem inside a simple polygon, as follows: Let $\overline{a_1b_1}, \ldots, \overline{a_tb_t}$ $(t \geqslant 1)$ be the edges of $\mathcal{P}$ shared by successive facets of F_i, where $\overline{a_1b_1}$ is closest to v_i and $\overline{a_tb_t}$ is closest to u_{i+1}. Note that these edges are necessarily vertical (with respect to $\boldsymbol{d}$). Consider the portion H_i of F_i which is enclosed between the chains $v_ia_1 \ldots a_tu_{i+1}$ and $v_ib_1 \ldots b_tu_{i+1}$ (see Fig. 6). From the triangle inequality, it is clear that any shortest (v_i, u_{i+1})-path lying in F_i must also lie in H_i. We can flatten H_i on the plane without changing edge lengths to get a simple polygon, which we continue to call H_i. (In fact, H_i is monotone in the direction $\boldsymbol{d}$.) Our problem now becomes one of finding a shortest (v_i, u_{i+1})-path in H_i, which we can do in $O(|H_i|)$ time using the algorithm in [11]. (This algorithm requires that H_i be triangulated in linear time. This can be done by simply adding the segments $\overline{a_ib_i}$ $(1 \leqslant i \leqslant t)$ and $\overline{a_ib_{i+1}}$ $(1 \leqslant i \leqslant t-1)$.)

3.4.1. Computing a shortest loop

An important special case arises if $B(\boldsymbol{d})$ consists entirely of $\boldsymbol{d}$-parallel facets. Let Q denote the circular sequence of these facets. Now, $L(\boldsymbol{d})$ is a shortest closed chain of line segments (i.e., *a loop*) lying in Q (see Fig. 7). However, it is not clear how to compute it since we do not have, at this point, a start vertex and an end vertex between which to run the shortest path algorithm—in general the shortest loop could potentially cross each of the shared vertical edges anywhere.

Fortunately, Lemma 3.4 below establishes that there is always a shortest loop which passes through a certain vertex of Q. Let $\overline{a_1b_1}, \ldots, \overline{a_tb_t}$ $(t \geqslant 1)$ be the vertical edges shared by successive facets of Q, where b_i is below a_i for all i. Let A (respectively B) be the closed chain of edges $\overline{a_1a_2}, \ldots, \overline{a_ta_1}$ (respectively $\overline{b_1b_2}, \ldots, \overline{b_tb_1}$). Let Q' be the portion of Q lying between A and B. By the triangle inequality, it follows that any shortest loop lying in Q must also lie in Q'.

Lemma 3.4. *There is a shortest loop around Q which passes through the lowest vertex, a_ℓ, of A.*

Proof. Let J be any shortest loop around Q—hence J lies in Q'. We first claim that if J bends upwards (respectively downwards), then it can do so only at a vertex of A (respectively B). Why? Assume that J bends upwards at a point p. (The discussion is similar if J bends downwards.) For some i, p lies in the trapezoid of Q' defined by the vertices a_i, a_{i+1}, b_i, and b_{i+1}. If p is any point of this trapezoid other than a_i and a_{i+1}, then we can shorten J by picking two points on it that are sufficiently close to p and

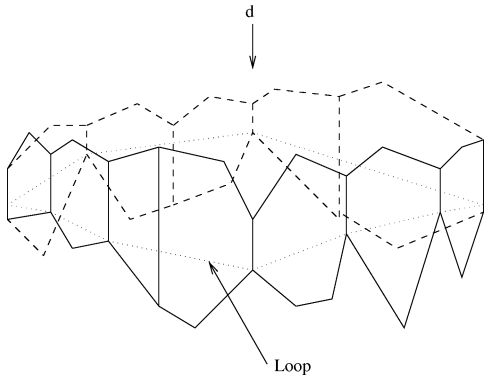

Fig. 7. A shortest loop around a circular sequence of $\boldsymbol{d}$-parallel facets.

on opposite sides of the vertical line through p and joining them directly while still staying within Q' (hence within Q). This contradicts the optimality of J and establishes the claim.

For the rest of this proof we take J to be a highest possible shortest loop lying in Q', i.e., we take a shortest loop around Q' and push it upwards as far as possible without leaving Q'. By the above claim, it follows that J passes through at least one vertex of A. Let a_ℓ be the lowest vertex of A and assume, for a contradiction, that J does not pass through a_l. Let $c \neq a_l$ be the point where J crosses $\overline{a_l b_l}$. At c, J has three possible orientations: it is (i) horizontal, (ii) sloping downwards to the right, or (iii) sloping upwards to the right.

Case (i). Since J must pass through a vertex of A and this vertex cannot be lower than a_l, it follows that as we walk along J to the right of c, J must bend upwards. Consider the first bend. By the claim above, this bend (or any other upward bend for that matter) can occur only at some vertex of A. But then this vertex is lower than a_ℓ since J is horizontal at c and c is below a_l—a contradiction.

Case (ii). As we walk along J to the right of c, we move downwards. Since J is a loop it must bend upwards at some point in order to return to c. Consider the first such bend. By the above claim, this bend must be at a vertex of A. But then this vertex is lower than a_ℓ—again a contradiction.

Case (iii). Similar to case (ii), except that we walk along J to the left of c. □

Lemma 3.4 gives us the desired start and end vertex for the shortest path algorithm. Specifically, we cut Q' along $\overline{a_l b_l}$ and flatten it out into a polygon in the plane; thus $\overline{a_l b_l}$ appears at the two ends of the polygon. We then run the algorithm of [11] between the two copies of a_ℓ to find the shortest loop. This takes $O(|Q'|)$ time.

3.5. Putting it all together: the overall algorithm and its analysis

In Sections 3.2 and 3.3, we showed how to set up and solve the optimization problem for the interior of each cone C (respectively edge e), assuming that the parting line $L(C)$ (respectively $L(e)$) was given. Similarly, in Section 3.4, we showed how to compute for each vertex, $\boldsymbol{d}$, the shortest path within each group, F_i, of $\boldsymbol{d}$-parallel facets, assuming that $B(\boldsymbol{d}) = c_1 F_1 \ldots c_m F_m$ was given. In this section, we show how to compute $L(C)$, $L(e)$, and $B(\boldsymbol{d})$ in $O(n)$ time. This immediately implies an $O(n^3)$-time algorithm

for finding the flattest parting line. In Section 3.6, we will describe an incremental approach which brings the time down to $O(n^2)$.

We assume that $\mathcal{P}$ is represented by doubly-connected edge lists [16], so that common operations such as walking around a facet, determining adjacent facets, etc. can be done efficiently. We first show how to compute $\mathcal{A}$.

3.5.1. Computing $\mathcal{A}$

Let $\mathbb{S}^2_+$ be the upper hemisphere of $\mathbb{S}^2$. Using a mapping called *central projection* [16], which establishes a bijection between points on $\mathbb{S}^2_+$ and points in the plane, we map the portions of the great circles lying in $\mathbb{S}^2_+$ to straight lines in the plane. Using the algorithm of [9], we compute the arrangement of these lines in $O(n^2)$ time. The faces, edges, and vertices of this arrangement are in 1–1-correspondence with the cones, edges, and vertices of the portion $\mathcal{A}^+$ of $\mathcal{A}$ lying in $\mathbb{S}^2_+$. Thus, by inverting the mapping we can compute $\mathcal{A}^+$. We repeat the above for the lower hemisphere of $\mathbb{S}^2$ also.

3.5.2. Computing $L(C)$

Let $\boldsymbol{d}$ be an interior point of C. We compute $F(\boldsymbol{d})$ in $O(n)$ time by testing each facet of $\mathcal{P}$ for $\boldsymbol{d}$-visibility in $O(1)$ time. A facet $f \in F(\boldsymbol{d})$ is a *boundary facet* (i.e., it contributes to $B(\boldsymbol{d})$) iff at least one facet f' adjacent to f is not in $F(\boldsymbol{d})$. If f is a boundary facet, then its contribution to $B(\boldsymbol{d})$ is the sequence of edges in $f \cap f'$, for all f' as above. We can determine whether f is a boundary facet and, if so, its contribution to $B(\boldsymbol{d})$ by walking around f and testing each adjacent facet. This takes $O(|f|)$ time per f, hence $O(n)$ time in total. We then concatenate the sequences contributed by the different boundary facets into a circular doubly-linked list representing $L(C)$. To do this, we observe that an endpoint (vertex) of a sequence is also the endpoint of exactly one other sequence. We can thus use an array indexed by vertices to determine the order in which to connect the sequences together. This takes $O(n)$ time.

Having thus constructed $L(C)$ in $O(n)$ time, we proceed to set up and solve the optimization problem for C's interior in $O(n)$ additional time, as discussed in Sections 3.2 and 3.3. Thus, all cones can be processed in $O(n^3)$ time.

3.5.3. Computing $L(e)$

We pick a point $\boldsymbol{d}$ in e's interior and compute $L(e)$ in much the same way we computed $L(C)$. The only difference is that now there can be up to two $\boldsymbol{d}$-parallel facets, f_1 and f_2, in $B(\boldsymbol{d})$ in addition to the computed sequences c_1 and c_2 (see Fig. 3). To get $L(e)$, we include the segments $\overline{v_1u_2} \in f_1$ and $\overline{v_2u_1} \in f_2$. Thereafter, we formulate and solve the optimization problem as before. The total time for processing all the edges is also $O(n^3)$.

3.5.4. Computing $B(\boldsymbol{d})$ and $L(\boldsymbol{d})$ for a vertex $\boldsymbol{d}$

We compute $L(\boldsymbol{d})$ as follows. Using our earlier notation (Section 3.4), $L(\boldsymbol{d}) = c_1 p_1 \ldots c_m p_m$, where c_i and c_{i+1} are chains of edges of $\mathcal{P}$ separated by a group, F_i, of contiguous $\boldsymbol{d}$-parallel facets and p_i is a shortest path in F_i connecting these two chains. We first show how to compute $B(\boldsymbol{d}) = c_1 F_1 \ldots c_m F_m$.

As in the case for cones, $c_1, \ldots, c_m$ can be computed in $O(n)$ time. Also the $\boldsymbol{d}$-parallel facets can be found in $O(n)$ time. What remains is to group these facets into $F_1, \ldots, F_m$ and then form the circular list $c_1 F_1 \ldots c_m F_m$. For each $\boldsymbol{d}$-parallel facet f, we scan f and determine its vertical edges (with respect to $\boldsymbol{d}$; these are edges of the type $\overline{ab}$ in Section 3.4). This takes $O(|f|)$ time per f, hence $O(n)$ time in all. There can be zero, one, or two such edges in f. If there are zero such edges, then f forms an F_i

by itself. Otherwise, f is contiguous with a $\boldsymbol{d}$-parallel facet along each of its vertical edges. Since any two $\boldsymbol{d}$-parallel facets share at most one vertical edge, we can use an array indexed by edges to determine the order in which to connect the facets to form the groups $F_1, \ldots, F_m$. This takes $O(n)$ time. The list $c_1 F_1 \ldots c_m F_m$ can be formed in $O(n)$ additional time by storing the vertices of all the F_i's in an array and indexing into this array using the endpoints of each c_j.

Having found $B(\boldsymbol{d})$, we can compute the shortest path p_i in F_i by constructing the polygon $H_i \subseteq F_i$ and running the shortest path algorithm of [11] within each H_i (see Section 3.4). This gives $L(\boldsymbol{d})$. (The discussion when $L(\boldsymbol{d})$ is a shortest loop around $\mathcal{P}$ is similar and hence omitted.)

Excluding the time for computing the different H_i's and p_i's, the time taken to compute $L(\boldsymbol{d})$ is $O(n)$. (We will show below that the time to construct the H_i's and p_i's that arise during the processing of *all* the vertices of G is just $O(n)$.) Given $L(\boldsymbol{d})$ we can compute $\widehat{L}(\boldsymbol{d})$ and then $\rho(\boldsymbol{d})$ in $O(n)$ time. It follows that all the vertices can be processed in $O(n^3)$ time.

Let us now obtain an upper bound on the time to compute all the H_i's and p_i's. Recall that for each F_i, we have already computed the vertical edges of the facets comprising F_i. Therefore, we can compute H_i in $O(|F_i|)$ time, where $|F_i|$ is the number of facets in F_i. The shortest path algorithm on H_i takes $O(|H_i|)$ time. We charge this time equally to the $\Theta(|H_i|)$ vertical edges of H_i, which results in a charge of $O(1)$ to each vertical edge $\overline{ab}$. How many times is $\overline{ab}$ charged in this way over the whole phase? $\overline{ab}$ is charged whenever it is vertical with respect to a direction. For any edge, there are exactly two directions in which it is vertical. Thus $\overline{ab}$ is charged only $O(1)$ over the whole phase. Since $\mathcal{P}$ has $O(n)$ edges, it follows that the time to build all the H_i's and p_i's is $O(n)$.

Thus we have the following theorem.

Theorem 3.1. *A flattest, undercut-free parting line for an n-vertex convex polyhedron, based on Criterion ρ, can be computed in $O(n^3)$ time.*

3.6. Speeding up the algorithm: incremental computation of parting lines

The bottleneck in the above approach is the computation of each parting line from scratch, at a cost of $O(n)$ apiece. We now show how to sequence these computations so that each parting line can be computed incrementally from the previous one, at a total cost of $O(n^2)$. As seen already, the rest of the algorithm—formulating and solving the optimization and shortest path problems—also takes $O(n^2)$ time. Thus, the entire algorithm runs in $O(n^2)$ time.

The computation proceeds in two phases. In the first phase, the parting lines for the interiors of the cones and edges are computed. In the second phase, the parting lines for the vertices are computed.

3.6.1. The first phase

Let $\overline{\mathcal{A}}$ be the dual graph of $\mathcal{A}$, obtained by placing a vertex $\overline{C}$ inside each cone C and joining two vertices of $\overline{\mathcal{A}}$ by an edge $\overline{e}$ if the corresponding cones share an edge e. We will process cones and edges in the order in which their duals are encountered in a depth-first search (dfs) of $\overline{\mathcal{A}}$.

We pick a cone C and compute $L(C)$ in $O(n)$ time, as in Section 3.5.2. We then perform a dfs of $\overline{\mathcal{A}}$ starting at $\overline{C}$. Assume that the search next visits vertex $\overline{C}'$ via edge $\overline{e}$. That is, in $\mathcal{A}$ we go from C to C' by crossing edge e. We compute $L(e)$ from $L(C)$ as follows.

Let $\boldsymbol{d}$ be a point in e's interior. Recall that there are at most two $\boldsymbol{d}$-parallel facets, f_1 and f_2, associated with e (Fig. 5). Assume that both exist. One of them (say, f_1) is not visible from the interior of C while

the other one (f_2) is visible. As we move to $\boldsymbol{d}$ from the interior of C, f_1 will start coming into view while f_2 will begin to disappear. Each facet f which is adjacent to f_1 and is a boundary facet in C will cease to be a boundary facet now, while f_1 will become one. Therefore, we delete f's contribution, namely $f \cap f_1$, from $L(C)$. Also, we delete f_2's contribution from $L(C)$ since it is now $\boldsymbol{d}$-parallel. The net result is that $L(C)$ is split into the two chains c_1 and c_2. To get $L(e)$, we add in the segments $\overline{v_1u_2}$ and $\overline{v_2u_1}$.

The time taken to process f_1 and f_2 in this way is $O(|f_1| + |f_2|)$. Let us upper-bound the total time it takes to process f_1 and f_2 over the entire phase. Let g be the great circle containing e. Note that f_1 and f_2 will appear as $\boldsymbol{d}$-parallel facets whenever $\boldsymbol{d}$ lies in the interior of an edge of g. Since there are $O(n)$ edges contained in g, the total time for processing f_1 and f_2 is $O(n(|f_1|+|f_2|))$. Now let us upper-bound the total time to handle all pairs of such $\boldsymbol{d}$-parallel facets as we visit edges on other great circles—this is the time it takes to incrementally compute the parting lines for all edges. By summing the above bound over all great circles and noting that each facet of $\mathcal{P}$ corresponds to exactly one great circle, we see that the total time is

$$O\big(n\big(|f_1| + |f_2| + |f_3| + \cdots\big)\big) = O\bigg(n \sum_{f\in\mathcal{P}} |f|\bigg) = O\big(n^2\big).$$

We can now construct $L(C')$ from $L(e)$ by essentially reversing the above process. The time to compute $L(C')$ will be $O(|f_1| + |f_2|)$. Arguing as above, the time to compute the parting lines for all cones is $O(n^2)$.

3.6.2. The second phase

Here we process vertices of $\mathcal{A}$ in the order that they are encountered in a dfs of $\mathcal{A}$ (not $\overline{\mathcal{A}}$). The phase is initialized by selecting a vertex $\boldsymbol{d}$ of $\mathcal{A}$, computing $L(\boldsymbol{d})$ in $O(n)$ time as discussed in Section 3.5.4, and then doing a dfs of $\mathcal{A}$ from $\boldsymbol{d}$.

Suppose that the dfs visits vertex $\boldsymbol{d}'$ from vertex $\boldsymbol{d}$, along edge $\overline{\boldsymbol{d}\boldsymbol{d}'}$. How does $L(\boldsymbol{d})$ change to $L(\boldsymbol{d}')$? As we move from $\boldsymbol{d}$ to $\boldsymbol{d}'$, some of the $\boldsymbol{d}$-parallel facets will start to come into view while others start disappearing from view; this will cause some of the F_i's to be replaced by chains of edges. Also, some facets will become $\boldsymbol{d}'$-parallel, causing some of the c_j's to be replaced by groups of contiguous $\boldsymbol{d}'$-parallel facets. Thus, $c_1F_1\ldots c_mF_m$ changes to $c'_1F'_1\ldots c'_sF'_s$ (for some integer s). We then run the shortest path algorithm within the H'_i corresponding to each F'_i and compute p'_i. This gives $L(\boldsymbol{d}') = c'_1p'_1\ldots c'_sp'_s$.

We discuss the computation of $c'_1F'_1\ldots c'_sF'_s$ from $c_1F_1\ldots c_mF_m$ in more detail now. Let h be the great circle containing $\boldsymbol{d}$ and $\boldsymbol{d}'$. Let g be a great circle intersecting h at $\boldsymbol{d}$ and let f_1 and f_2 be the (up to two) $\boldsymbol{d}$-parallel facets corresponding to g. Define g', f'_1 and f'_2 similarly with respect to $\boldsymbol{d}'$.

For each g intersecting h at $\boldsymbol{d}$, we do the following: Assume without loss of generality that f_1 is $\boldsymbol{d}'$-visible and f_2 is not. We delete f_1 from its F_i and add its lower boundary (with respect to $\boldsymbol{d}'$) in its place, since f_1 is a boundary facet with respect to $\boldsymbol{d}'$. Also, we delete f_2 from its F_j and replace it with the sequence $f \cap f_2$ for each facet f that is adjacent to f_2 and is not a boundary facet with respect to $\boldsymbol{d}$, since each such f now becomes a boundary facet with respect to $\boldsymbol{d}'$.

For each g' intersecting h at $\boldsymbol{d}'$ we do the following: Both f'_1 and f'_2 are $\boldsymbol{d}'$-parallel. Assume without loss of generality that f'_1 is $\boldsymbol{d}$-visible and f'_2 is not; f'_1 will contribute to some chain c_j of $L(\boldsymbol{d})$, while f'_2 does not contribute to any chain. We delete f'_1's contribution from $L(\boldsymbol{d})$ and insert f'_1 in its place. We also delete $f \cap f'_2$ for each facet f which is adjacent to f'_2 and is a boundary facet with respect to $\boldsymbol{d}$ and we insert f'_2 in its place. (In this way, f'_1 and f'_2 become a part of some F'_i and some F'_j, respectively.) Also we scan f'_1 and f'_2 and compute their vertical sides (with respect to $\boldsymbol{d}'$).

At this point we have obtained $c'_1F'_1 \ldots c'_sF'_s$. We then construct $H'_i \subseteq F'_i$ and run the shortest path algorithm within it to get p'_i. This gives $L(\boldsymbol{d}')$.

How much time does all this take? (Again, let us exclude the time for constructing the H'_i and the p'_i; we saw in Section 3.5.4 that all such computations take only $O(n)$ time.) The time to process f_1 and f_2 as above is $O(|f_1| + |f_2|)$. Now, f_1 and f_2 will be processed in this way at each of the $O(n)$ vertices created by the intersection of the great circle g with another great circle. Similarly for f'_1 and f'_2. Thus, the total time to process all the vertices of $\mathcal{A}$ is $O(n \sum_{f\in\mathcal{P}} |f|) = O(n^2)$.

We may now conclude the following.

Theorem 3.2. *Using the incremental approach to update parting lines, a flattest, undercut-free parting line for an n-vertex convex polyhedron, based on Criterion ρ, can be computed in $O(n^2)$ time.*

4. Implementation and discussion

We have implemented a version of the algorithm just described, excluding the incremental computation. Our goal was to compare the best and worst parting lines for a "typical" convex polyhedron. Figs. 8 and 9 show these (in heavy lines) for an example 40-vertex convex polyhedron, where the best line has $\rho = 0.9452$ and the worst line has $\rho = 0.2980$. The polyhedron was generated using `qhull` (`http://www.geom.umn.edu/software/download/qhull.html`) to compute the convex hull of forty co-spherical points generated randomly. Our implementation is written in C++, runs on an SGI Irix 5 machine and uses LEDA (`http://www.mpi-sb.mpg.de/LEDA`) to compute $\mathcal{A}$ and the shortest paths.

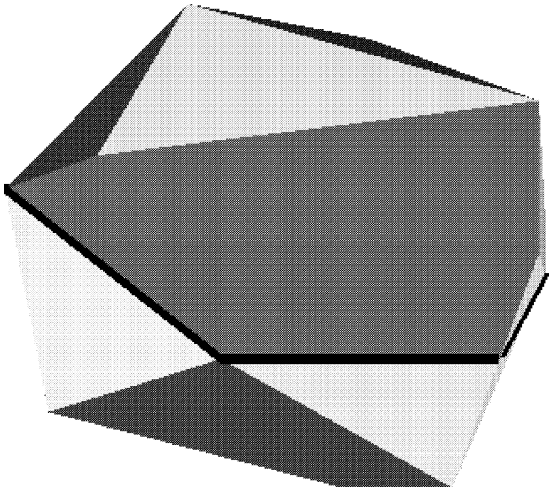

Fig. 8. Best parting line produced under Criterion ρ. Here $\rho = 0.9452$.

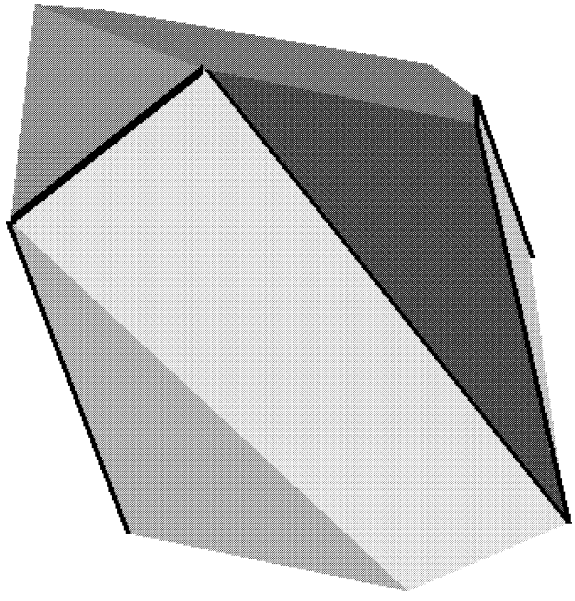

Fig. 9. Worst parting line produced under Criterion ρ. Here $\rho = 0.2980$.

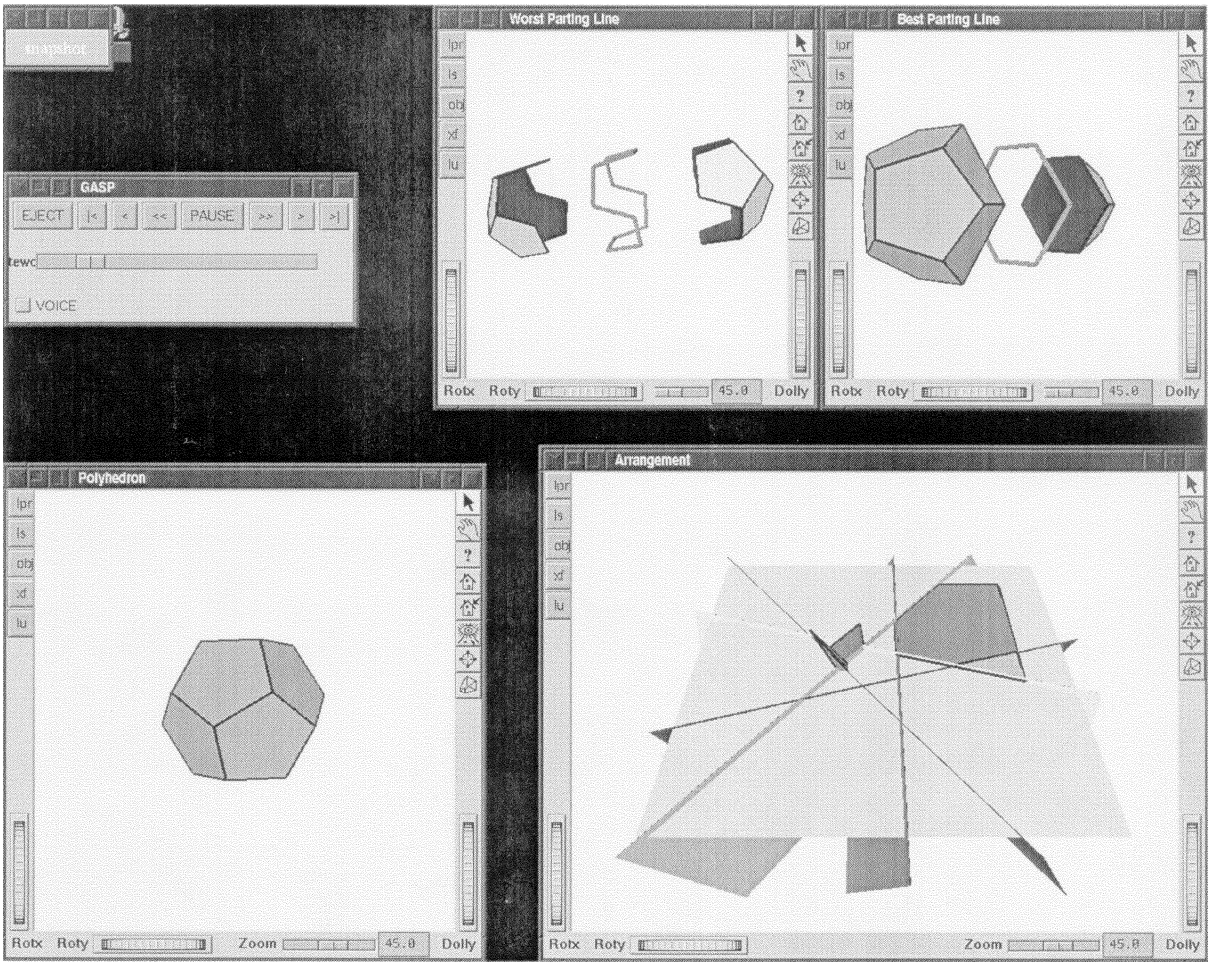

Fig. 10. A frame from the animation of the algorithm.

Even without the incremental computation, the implementation runs very fast—for the 40-vertex polyhedron, it takes only 0.16 seconds to compute the optimal parting line (excluding the time needed for graphical output).

We have also animated our algorithm using the GASP system [19] (available via anonymous FTP from `ftp.cs.princeton.edu, path people/ayt/gasp.tar.Z`). Fig. 10 is a frame from the animation. It depicts a sample polyhedron $\mathcal{P}$ (a dodecahedron), the arrangement $\mathcal{A}$ (shown in the plane), and exploded views of the best and worst parting lines for $\mathcal{P}$. Note that the best parting line lies completely in a plane, as one would expect in this case, while the worst parting line is highly stepped. The darkly-shaded regions in $\mathcal{A}$ indicate, respectively, the face, edge, and vertex where the best parting line for a cone-interior, an edge-interior, and a vertex were found. (For clarity, the region for a vertex is shown enhanced as a wedge.) The lightly-shaded triangle shows where the worst parting line was found.

5. Flattest undercut-free parting line under Criterion ρ'

Recall from Section 2.2 that Criterion ρ' is given by

$$\rho'(\boldsymbol{d}) = \sum_{i=1}^{k} \text{length}(\widehat{e}_i) \Big/ \sum_{i=1}^{k} \text{length}(e_i). \tag{3}$$

The only change that needs to be made to the algorithm of Section 3 is in the optimization problem. For the interior of a cone C, the optimization problem is

$$\begin{aligned}
&\text{maximize} \quad f(x,y,z) = \sum_{\ell=1}^{k} \sqrt{A_\ell x^2 + B_\ell y^2 + C_\ell z^2 + D_\ell xy + E_\ell yz + F_\ell xz} \\
&\text{subject to} \quad x^2 + y^2 + z^2 = 1 \quad \text{(sphere constraint)}, \\
&\qquad \boldsymbol{d} \cdot \boldsymbol{n}_f > 0 \text{ (respectively } \boldsymbol{d} \cdot \boldsymbol{n}_f < 0) \text{ if } \boldsymbol{d}_C \cdot \boldsymbol{n}_f > 0 \text{ (respectively } \boldsymbol{d}_C \cdot \boldsymbol{n}_f < 0) \\
&\qquad\quad \text{for each great circle } g_f \text{ bounding } C \quad \text{(plane constraints)}.
\end{aligned}$$

This optimization problem is cumbersome to solve with the Lagrangian Multipliers method because of the square roots. Instead, in our implementation, we used the optimization function `constraint` provided in the MATLAB package.

Let $T(n)$ denote the time it takes to solve an instance of such a problem. ($T(n)$ will, of course, depend on the algorithm used by the optimization function, e.g., `constraint`. In the worst case, $T(n) = \Omega(n)$, since the size, k, of the objective function can be $\Theta(n)$.) Then the total time, taken over all cones, is $O(n^2 T(n))$. Additionally, we need to update the objective function as we move from one cone to the next. The size of the objective function here is proportional to that of the parting line. As the example below shows, it is possible for the sum of the sizes of the parting lines, taken over all cones, to be $\Theta(n^3)$. Thus, it does not help to use the incremental approach here. Instead, we simply compute the objective function afresh at each cone, for a total cost of $O(n^3)$. Thus, the running time of the algorithm is $O(n^3 + n^2 T(n))$. We handle edge interiors and vertices analogously.

Example. Let $\mathcal{P}$ be a vertical pyramid, formed by joining the vertices of a regular n-gon (n even) to a vertex, v, that is located directly above the center of the polygon. Consider the facets f of P, excluding the base. Clearly, the arrangement of the planes, h_f, on the upper hemisphere of $\mathbb{S}^2$ has $\Theta(n^2)$ cones. For any direction $\boldsymbol{d}$ within one of these cones, at least $n/2$ facets of $\mathcal{P}$ are visible. (To see this, consider facets f and f' determined by two parallel edges of the regular n-gon. Then, from any direction $\boldsymbol{d}$ in the upper hemisphere of $\mathbb{S}^2$, at least one of f and f' is visible.) Since the base of $\mathcal{P}$ is not visible from $\boldsymbol{d}$, the edges shared by the visible facets and the base will form a portion of the parting line. Therefore, the parting line has size $\Theta(n)$ and it follows that the total size of the parting lines taken over all cones is $\Theta(n^3)$.

Figs. 11 and 12 depict the best and the worst undercut-free parting lines, under Criterion ρ', for the 40-vertex convex polyhedron used earlier. For the best (respectively worst) parting line, $\rho' = 0.9562$ (respectively $\rho' = 0.5229$).

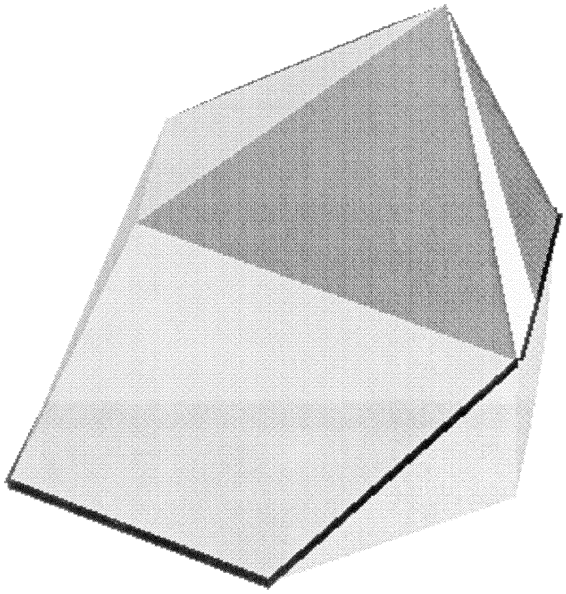

Fig. 11. Best parting line produced under Criterion ρ'. Here $\rho' = 0.9562$.

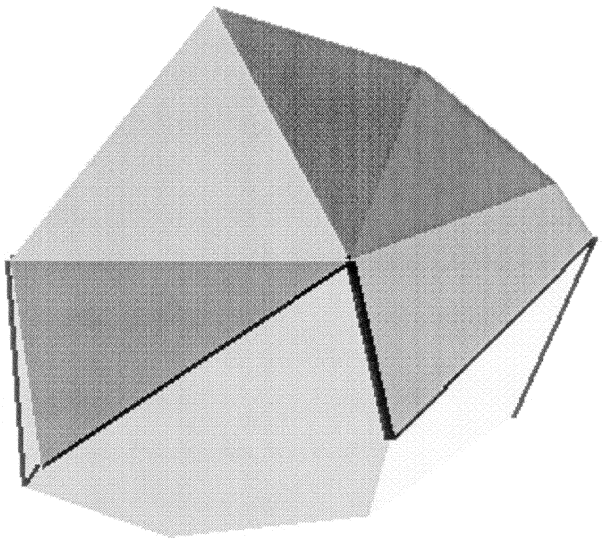

Fig. 12. Worst parting line produced under Criterion ρ'. Here $\rho' = 0.5229$.

6. Flattest undercut-free parting line under Criterion ω

Recall that the width, $\omega(\boldsymbol{d})$, of $L(\boldsymbol{d})$ in direction $\boldsymbol{d}$ is the smallest distance between two parallel planes that are perpendicular to $\boldsymbol{d}$ and enclose $L(\boldsymbol{d})$. To find a $\boldsymbol{d}$ which minimizes $\omega(\boldsymbol{d})$, we partition 3-space into cones as before and compute separately the width-minimizing direction for the parting line corresponding to each region (i.e., cone interior, edge interior, and vertex). The main difficulty is that the parting line changes as we move from region to region. Therefore, we need to restrict our search for the width-minimizing direction to the appropriate region. Additionally, at vertices we need to handle parallel facets appropriately, by finding a minimum-width parting line which passes through the parallel facets (Section 2.3).

Our approach to computing the width is based on [12]. Towards this end, we review some useful ideas from [12]. Let S be a point-set in 3-space and let $\mathrm{CH}(S)$ be its convex hull. Vertices u and v of $\mathrm{CH}(S)$ are called an *antipodal vertex–vertex pair* (or *VV pair*) if there exist parallel planes, one containing u and the other containing v, that enclose $\mathrm{CH}(S)$. Similarly, one can define *antipodal vertex–edge* (*VE*), *vertex–face* and *edge–edge* (*EE*) *pairs*. (Edge–face and face–face pairs are subsumed by vertex–face pairs, and hence not considered.)

In [12] it is shown that the width-minimizing direction for S is perpendicular to the parallel planes associated with VF or an EE pair. (The other cases do not sufficiently constrain the parallel planes and one can always find a direction in which to rotate so as to minimize the width.) We refer the reader to [12] for more details.

6.1. Minimum-width parting line for the interior of a cone

Let C be a cone and let $L(C)$ be the parting line for the interior of C. The following lemma characterizes the width-minimizing direction $\boldsymbol{d}$ for $L(C)$, where, for convenience in the proof, we let $\boldsymbol{d}$ range over the interior of C as well as its boundary.

Lemma 6.1. *The width-minimizing direction $\boldsymbol{d}$ for $L(C)$ satisfies one of the following*:
(i) *it is perpendicular to the parallel planes associated with an antipodal VF or EE pair of* $\mathrm{CH}(L(C))$ *that lies in the interior of C*;
(ii) *it lies on the boundary of C*.

Proof. If $\boldsymbol{d}$ points in any other direction, then it is perpendicular to the planes associated with a VV or VE pair lying in the interior of C. But then, one can find a direction in which to rotate these planes so as to reduce the width [12]. □

However, we can exclude the directions specified by Lemma 6.1(ii) because we will later compute the minimum-width parting lines for the edges and vertices on the boundary of C and the width of these will be no greater than that of $L(C)$ at the excluded directions. This is because $L(C)$ can also be used as an undercut-free parting line at the boundary of C and so the best undercut-free parting line for the boundary can be no worse than $L(C)$.

Let I be the number of directions given by Lemma 6.1(i) and let h denote the size of the parting line. Then the width-minimizing direction for the interior of C can be found in $O(h \log h + I)$ time [12]. Since I can be $\Theta(h^2)$ and h can be $\Theta(n)$, we spend $O(n^2)$ time per cone, hence $O(n^4)$ time for all cones.

6.2. Minimum-width parting line for the interior of an edge

Let e be an edge bounding C. Within the interior of e, we can have up to two $\boldsymbol{d}$-parallel facets as shown in Fig. 5. It is clear that any two parallel planes that enclose c_1 and c_2, will also enclose the segments $\overline{u_1v_2}$ and $\overline{u_2v_1}$. So, without loss of generality we can assume that the parting line, $L(e)$, for the interior of e consists of $c_1, c_2, \overline{u_2v_1}$, and $\overline{u_1v_2}$. The following lemma characterizes the width-minimizing direction $\boldsymbol{d}$ for $L(e)$, where we let $\boldsymbol{d}$ range over the interior of e as well as its bounding vertices.

Lemma 6.2. *The width-minimizing direction $\boldsymbol{d}$ for $L(e)$ satisfies one of the following*:
(i) *it is perpendicular to the parallel planes associated with an antipodal VF, EE, or VE pair of* $\mathrm{CH}(L(e))$ *that lies in the interior of e*;
(ii) *it corresponds to one of the vertices bounding e*.

Proof. If $\boldsymbol{d}$ points in any other direction, then it is perpendicular to the planes associated with a VV pair of $\mathrm{CH}(L(e))$. But in this case the width can be reduced by rotating the planes in the direction given by e. □

Again, we can exclude the two directions corresponding to the bounding vertices of e as we will later compute the minimum-width undercut-free parting lines for these directions. These lines will be no worse than $L(e)$ at the excluded directions, since $L(e)$ is also an undercut-free parting line at the bounding vertices of e.

The VF and EE pairs specified in Lemma 6.2(i) give us a discrete set of points (directions) on $\mathbb{S}^2$. However, each VE pair in Lemma 6.2(i) gives us a great circle of directions (since a pair of planes through a vertex and an edge can be rotated a full 360°). But, we are only interested in directions that are also in the interior of e. So, for each VE pair we find the great circle of directions, intersect it with e, and consider only the intersection point as a candidate direction. It follows that the best direction among these discrete points can be found in $O(h \log h + I) = O(n^2)$ time. Thus, the total time spent over the edges is $O(n^4)$.

6.3. Minimum-width parting line for a vertex

Let $v = \boldsymbol{d}$ be a bounding vertex of a cone C. In this case, we could have several $\boldsymbol{d}$-parallel facets, as shown in Fig. 3. Let us now define p_i to be any path lying in F_i and joining v_i and u_{i+1}. Clearly, the chain $c_1 p_1 c_2 p_2 \ldots c_m p_m$ is an undercut-free parting line with respect to de-molding in directions $\boldsymbol{d}$ and $-\boldsymbol{d}$. Among all such parting lines, we seek the one with the minimum width in direction $\boldsymbol{d}$. Let V be the following set of line segments:

(i) Edges that are shared by two adjacent $\boldsymbol{d}$-parallel facets. These edges are vertical (with respect to $\boldsymbol{d}$).
(ii) Vertices of all the chains c_i. We think of these vertices as degenerate vertical segments. (These may not exist if we have a loop, as in Fig. 7.)

Lemma 6.3. *At a vertex $\boldsymbol{d}$, there exists an undercut-free parting line $L(\boldsymbol{d})$ of width w iff there are parallel planes h_1 and h_2 that are distance w apart and perpendicular to $\boldsymbol{d}$, such that the region enclosed between h_1 and h_2 intersects all the vertical segments of V.*

Proof. We need only note that any undercut-free parting line for direction $\boldsymbol{d}$ must pass through at least one point of each of the vertical edges that are shared by two adjacent $\boldsymbol{d}$-parallel facets. □

By the above lemma it is clear that all we need to find is two parallel planes with minimum separation that are perpendicular to $\boldsymbol{d}$ and enclose at least one point of each of the vertical segments of V. Without loss of generality assume $\boldsymbol{d}$ points upwards along the positive z-axis. Each vertical segment has a lower and an upper endpoint. Let *low* be the highest lower endpoint and let *high* be the lowest upper endpoint. If $low \geqslant high$, then the region in between two planes that are normal to $\boldsymbol{d}$ and are at heights high and *low* will intersect all the vertical segments. If $low < high$ then any plane normal to $\boldsymbol{d}$ and at *height* between *low* and *high* will intersect all the vertical segments. Clearly, the minimum width is given by $\max(0, low - high)$. Therefore, we can find a minimum width undercut-free parting line in $O(n)$ time per vertex. So the total time over all the vertices is $O(n^3)$.

From the preceding discussion we may now conclude.

Theorem 6.1. *A flattest, undercut-free parting line for an n-vertex convex polyhedron, based on Criterion ω, can be computed in $O(n^4)$ time.*

7. Conclusion

We have given algorithms to compute, for a convex polyhedron, an undercut-free parting line which is as flat as possible. We have proposed two classes of flatness criteria—one based on the length of the

parting line and the other based on the positions of its vertices. Our methods include a combination of continuous optimization on the unit-sphere and discrete techniques from computational geometry. Our algorithms are general enough that other flatness criteria can be accommodated fairly easily. We have implemented some of our algorithms.

The obvious open problem is to handle non-convex polyhedra. This problem is considerably more complex, since there may not be a single direction which simultaneously minimizes the number of undercuts and also yields the flattest parting line. Therefore, one needs to formulate the problem in a way that reconciles these mutually conflicting requirements in a meaningful way. We are investigating this problem. Two potential approaches are: (i) the designer specifies thresholds for number of undercuts and flatness, and the goal is to find a direction for which both thresholds are met, or (ii) among all directions minimizing the number of undercuts find the direction yielding the flattest line (or vice versa).

Furthermore, there is a connectivity requirement that also must be incorporated into the solution. It is important that the parting line divides the polyhedron into a small number (ideally, two) of connected pieces. Otherwise, the cost of aligning and assembling the molds becomes prohibitive.

We view our results for the convex case as a first towards solving the non-convex problem, and we expect that the techniques and insights that we have gained will be useful in this regard.

Acknowledgements

We thank the Geometry Center at the University of Minnesota for providing access to computational and videotaping facilities during the implementation phase of this project. We also thank the referees for helpful feedback.

References

[1] H.-K. Ahn, M. de Berg, P. Bose, S.W. Cheng, D. Halperin, J. Matoušek, O. Schwartzkopf, Separating an object from its cast, in: Proceedings of the 13th ACM Symposium on Computational Geometry, 1997, pp. 221–230.
[2] R.W. Bainbridge, Thermo-setting plastic parts, in: J.G. Bralla (Ed.), Handbook of Product Design for Manufacturing, McGraw-Hill, New York, 1986, pp. 6-3–6-16.
[3] P. Bose, D. Bremner, M. van Kreveld, Determining the castability of simple polyhedra, in: Proceedings of the 10th Annual ACM Symp. on Computational Geometry, 1994, pp. 123–131.
[4] P. Bose, Geometric and computational aspects of manufacturing processes, Ph.D. Thesis, School of Computer Science, McGill University, Montréal, Canada, 1995.
[5] P. Bose, G. Toussaint, Geometric and computational aspects of gravity casting, Comput.-Aided Design 27 (6) (1995) 455–464.
[6] P. Bose, M. van Kreveld, G. Toussaint, Filling polyhedral molds, in: Proc. 3rd Workshop Algorithms Data Structures, Lecture Notes in Computer Science, Vol. 709, Springer, Berlin, 1993, pp. 210–221.
[7] L.-L. Chen, S.-Y. Chou, T.C. Woo, Parting directions for mould and die design, Comput.-Aided Design 25 (12) (1993) 762–768.
[8] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer, Berlin, 1987.
[9] H. Edelsbrunner, J. O'Rourke, R. Seidel, Constructing arrangements of lines and hyperplanes, with applications, SIAM J. Comput. 15 (1986) 341–363.
[10] S. Fekete, J. Mitchell, Geometric aspects of injection molding, Manuscript, 1992.

[11] L. Guibas, J. Hershberger, D. Leven, M. Sharir, R.E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, Algorithmica 2 (1987) 209–233.
[12] M.E. Houle, G.T. Toussaint, Computing the width of a set, IEEE Trans. Pattern Anal. Machine Intell. 10 (5) (1988) 761–765.
[13] K.C. Hui, S.T. Tan, Mould design with sweep operations – a heuristic search approach, Comput.-Aided Design 24 (2) (1992) 81–91.
[14] D.G. Luenberger, Introduction to Linear and Non-Linear Programming, Addison-Wesley, Reading, MA, 1973.
[15] M. McKenna, R. Seidel, Finding the optimal shadows of a convex polytope, in: Proceedings of the 1st Annual ACM Symp. on Computational Geometry, 1985, pp. 24–28.
[16] F.P. Preparata, M.I. Shamos, Computational Geometry – An Introduction, Springer, Berlin, 1988.
[17] B. Ravi, M.N. Srinivasan, Decision criteria for computer-aided parting surface design, Comput.-Aided Design 22 (1) (1990) 11–18.
[18] A. Rosenbloom, D. Rappaport, Moldable and castable polygons, in: Proceedings of the 4th Canadian Conference on Computational Geometry, 1992, pp. 322–327.
[19] A. Tal, D. Dobkin, GASP—a system to facilitate animating geometric algorithms, in: Proceedings of the 10th Annual ACM Symp. on Computational Geometry, 1994, 388–389.
[20] K. Tang, T. Woo, J. Gan, Maximum intersection of spherical polygons and workpiece orientation for 4- and 5-axis machining, J. Mech. Design 114 (1992) 477–485.
[21] E.C. Zuppann, Castings made in sand molds, in: J.G. Bralla (Ed.), Handbook of Product Design for Manufacturing, McGraw-Hill, New York, 1986, pp. 5-3–5-22.