Terrain Polygon Decomposition with Application to Layered Manufacturing

Ivaylo Ilinkin * Ravi Janardan* Michiel Smid †

February 25, 2002

Abstract

This paper considers the problem of decomposing a simple polygon into two terrain polygons. A polygon is a terrain polygon, if every boundary and interior point can be connected to an edge, called a base, by a perpendicular line segment contained entirely within the polygon. Algorithms are given that use $O(n)$ space and run in $O(n \log n)$ time, if the terrains have a common base, and in $O(n^2 \log n)$ time, otherwise. The problem is motivated by Layered Manufacturing, since such a decomposition allows the construction of certain classes of 3D parts very efficiently.

1 Introduction

Let $\mathcal{P}$ be a simple, n-vertex polygon, i.e., $\mathcal{P}$ is a subset of the plane bounded by a single closed (polygonal) chain of edges, where no two non-consecutive edges share a point. $\mathcal{P}$ is called a *terrain polygon* (or *terrain*, for short) if it has an edge, e, called a *base*, such that every point in $\mathcal{P}$ can be joined to e by a perpendicular line segment which lies entirely in $\mathcal{P}$ [FM01]. Figure 1 illustrates a terrain.

In this paper we consider two problems:

1. Decide if $\mathcal{P}$ is a terrain and, if it is, then find a base. We give a simple algorithm that runs in $O(n)$ time.

2. If $\mathcal{P}$ is not a terrain, then decide if it can be decomposed by a line into two terrains. If so, then compute a decomposing line and a base for each terrain. We give an algorithm which runs in $O(n \log n)$ time, if the terrains have a common base (on the decomposing line), and in $O(n^2 \log n)$ time, otherwise. Figure 2 illustrates a decomposition of $\mathcal{P}$ into two terrains.

The problem of decomposing a polygon into two terrains arises in Layered Manufacturing (LM), which is an emerging technology that allows physical prototypes of 3D solids to be created directly

*Department of Computer Science & Engineering and Army High Performance Computing Research Center, University of Minnesota, Minneapolis, MN 55455, U.S.A. Email: {ilinkin, janardan}@cs.umn.edu . Research supported, in part, by NSF grant CCR–9712226. This effort is also sponsored, in part, by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

†School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6. Email: michiel@scs.carleton.ca .

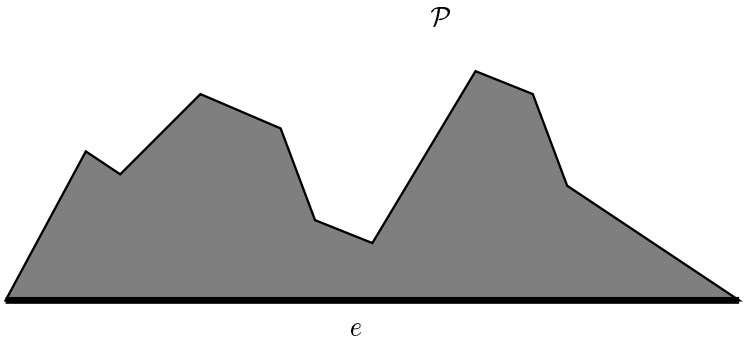

Figure 1: *A polygon $\mathcal{P}$ which is a terrain with base e.*

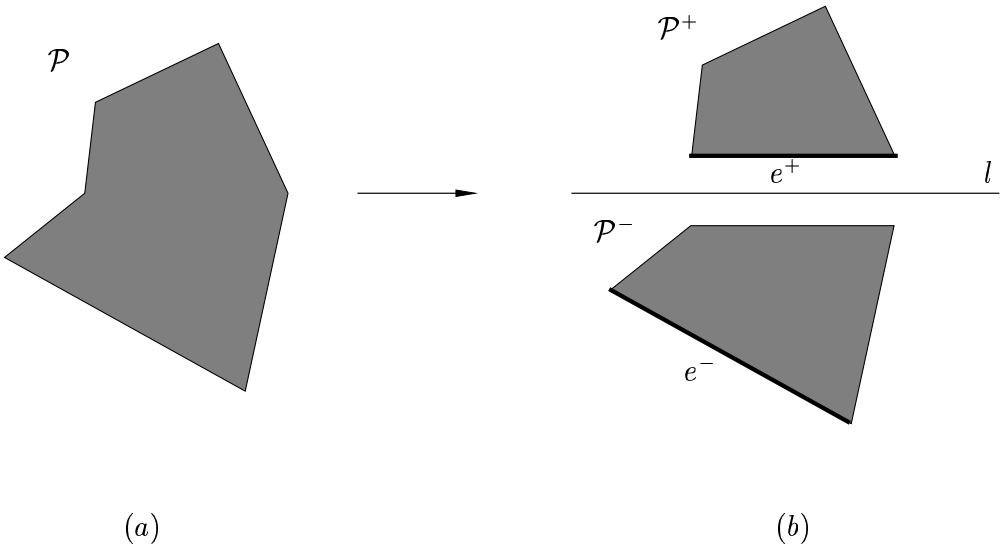

Figure 2: (a) *A polygon $\mathcal{P}$ which is not a terrain.* (b) *A decomposition of $\mathcal{P}$ by a line l into two terrains, $\mathcal{P}^+$ and $\mathcal{P}^-$, with bases e^+ and e^-, respectively.*

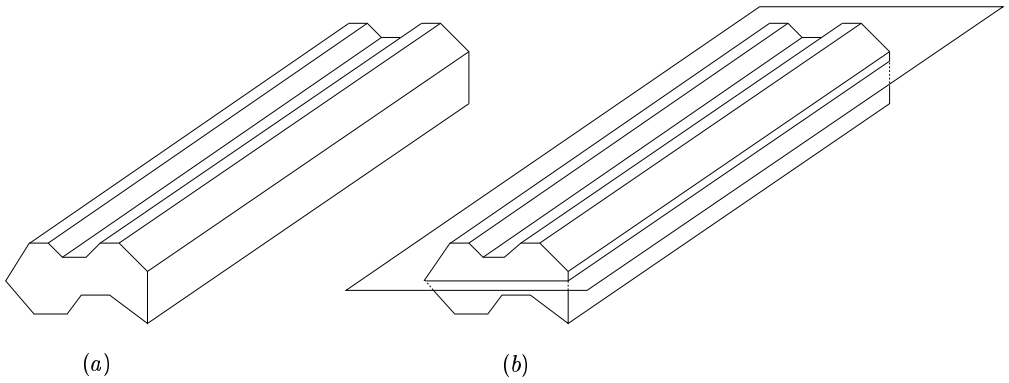

Figure 3: (*a*) *A long and slender object of uniform cross-section.* (*b*) *A decomposition of the object into two terrains with a common base.*

---

from their digital models. Briefly, LM works as follows (see [Jac92, KF98]): The 3D digital model (a polyhedron) is oriented suitably and sliced by a plane into parallel 2D layers. The layers (which are polygons) are shipped over a network to a fabrication device which "prints" them successively, each layer on top of the previous one. Successive layers adhere to each other, so that the 3D model is realized as a stack of 2D layers. The mechanics of the "printing" phase vary from process to process. In Stereolithography, it is accomplished by having a laser act on a light-sensitive liquid; in Fused Deposition Modeling, layers are printed by depositing fine strands of molten plastic, etc.

A key process-planning step in LM is the analysis of the model and its orientation to determine the need for *supports*. These are temporary structures that are generated as the model is built in order to prop-up portions of the model that do not have previously-built layers under them. Supports are undesirable as they increase the build time and material cost and degrade surface quality. Support requirements can be reduced by choosing the orientation of the model suitably [ABB$^+$97, MJS$^+$99, MJSG99, MJSS98].

The terrain decomposition problem comes up quite naturally in this setting. Consider building a part which is long and slender and has a uniform cross-section, e.g. a piston rod, a drive shaft, or a gun barrel (Figure 3(*a*) shows a generic such part). Orienting and building this part vertically (along the long axis) has the advantage that it requires no support. However, the process will be very slow due to the large number of layers that must be printed (layer thicknesses are typically 0.01", so a foot-long part could have over a thousand layers). Moreover, the stacking of so many layers and the high height-to-width ratio will lead to instabilities and inaccuracies in the part.

However, if the (polygonal) cross-section of the part is decomposable by a line into two terrains, then the part can still be built without supports, while avoiding the disadvantages mentioned above. We divide the 3D model into two pieces using a plane which contains the decomposing line and the long axis of the model, build each piece on its base facet (i.e., the facet containing the base edge), and then glue the pieces back together. The fact that the cross-section of each piece is a terrain implies that the piece can be built without supports (Figure 3(*b*)).

We remark that this approach also works if the cross-sections are not uniform but are scaled versions of one another (Figure 4). Such parts are often generated in CAD packages like AutoCAD, as solids of revolution, or by using extrusion features, or by combining similar-looking primitives

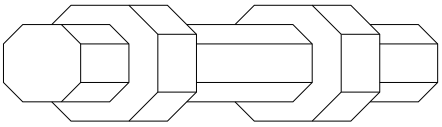

Figure 4: *An example of an object whose cross-sections are scaled versions of each other.*

---

in constructive solid geometry.

1.1 Related Work

The problem of deciding whether a polyhedron is a terrain has been considered in an equivalent form by Asberg *et. al.* [ABB+97] in the context of Stereolithography. They analyze the class of objects (modeled as polygons or polyhedra) that can be built without supports by *vertical stereolithography*, which is precisely the class of objects that are terrains.

In general it is natural to consider decomposing a polygon into the minimum number of terrains. However, Fekéte and Mitchell [FM01] have proven that it is NP-complete to decide if a polygon with holes or a polyhedron of genus zero can be decomposed into $k > 1$ terrains, where k is part of the input.

In related work, Ilinkin *et. al.* [IJM+01] have considered the following problem: given a polyhedron $\mathcal{P}$ and a direction $\mathbf{d}$, decompose $\mathcal{P}$ into two connected components, using a plane perpendicular to $\mathbf{d}$, such that the support requirement (support-volume or contact-area) is minimized when one component is built in direction $\mathbf{d}$ and the other in $-\mathbf{d}$. They give an algorithm which runs in $O(n \log n)$ time if $\mathcal{P}$ is convex and in $O(n^2 \log n)$ time otherwise.

Another related work is by Bose *et. al.* [BBvK97] in the context of castability. They consider the problem of decomposing a polyhedron by a plane and building mold-halves for the two pieces such that the cast polyhedron can be de-molded without obstruction. This is equivalent to deciding if the polyhedron can be decomposed into two polyhedra that are both terrains with respect to the common base formed by the dividing plane. We remark that their algorithms could be used to decide whether a simple polygon can be decomposed into two terrains with a common base. We present a more efficient solution and consider the general problem where the bases for the two pieces need not be the same.

The rest of the paper is organized as follows. In Section 2, we give a linear-time algorithm to decide if a simple polygon is a terrain and, if so, to compute a base. Similar results were first given in [ABB+97]. We give our results for completeness, since our algorithms in Section 3 and 4 rely on them. In Section 3, we give an $O(n \log n)$-time algorithm to decompose a simple polygon into two terrains with a common base, while in Section 4, we present an $O(n^2 \log n)$-time algorithm to decompose a simple polygon into two terrains that do not have a common base (if such decompositions exist). We conclude in Section 5.

2 Recognizing terrains

The following lemma gives a necessary and sufficient condition for a polygon $\mathcal{P}$ to be a terrain. Surprisingly, it depends only on the angles between edge normals; it does not depend on the location

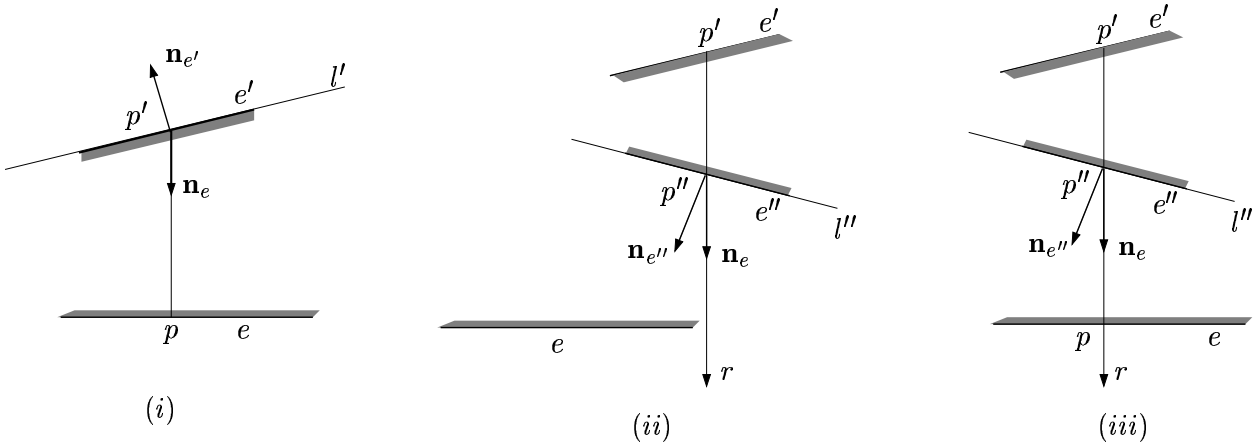

Figure 5: *Proof of Lemma 2.1.The shading of each edge indicates the interior of the polygon in the vicinity of the edge.*

---

or lengths of edges. Hereafter, we assume that all edge normals of $\mathcal{P}$ have been translated so that they begin at the origin. Furthermore, we will use the phrase "$\mathcal{P}(e)$ is a terrain" to mean that $\mathcal{P}$ is a terrain with base e.

Lemma 2.1 *A simple polygon $\mathcal{P}$ is a terrain if and only if there is an edge $e \in \mathcal{P}$ such that, for any other edge $e' \in \mathcal{P}$, $\mathbf{n}_e \cdot \mathbf{n}_{e'} \leq 0$, where $\mathbf{n}_e$ and $\mathbf{n}_{e'}$ are the outward-directed unit-normals of e and e', respectively.*

Proof

($\Rightarrow$) Assume that $\mathcal{P}$ is a terrain; denote its base by e. Let $e' \neq e$ be any other edge of $\mathcal{P}$. Thus, for every point $p' \in e'$ there is a point $p \in e$ such that $\overline{pp'}$ is perpendicular to e and lies inside $\mathcal{P}$ (see Figure 5(i)). Let l' be the line through e'. Translate the normals $\mathbf{n}_e$ and $\mathbf{n}_{e'}$ to p'. Then $\mathbf{n}_e$ points from p' to p and lies in the half-plane of l' that, in the vicinity of e', contains the interior of $\mathcal{P}$. The normal $\mathbf{n}_{e'}$ is perpendicular to e' and lies in the opposite half-plane. Therefore, $\mathbf{n}_e \cdot \mathbf{n}_{e'} \leq 0$.

($\Leftarrow$) Assume that $\mathcal{P}$ is not a terrain. Let e be any edge of $\mathcal{P}$. Thus, there must be an edge $e' \neq e$ and a point $p' \in e'$ such that the ray r originating at p' in the direction of $\mathbf{n}_e$ either (a) misses e or (b) meets e at a point p but $\overline{pp'}$ is not contained in $\mathcal{P}$.

Case (a): Let r leave $\mathcal{P}$ for the first time at a point p'' on an edge $e'' \neq e$ (see Figure 5(ii)); note that $p' \equiv p''$, possibly. Translate the normals $\mathbf{n}_e$ and $\mathbf{n}_{e''}$ to p''. Let l'' be the line through e''. Then $\mathbf{n}_e$ points from p'' into the open half-plane of l'' that, in the vicinity of e'' does not contain the interior of $\mathcal{P}$. The normal $\mathbf{n}_{e''}$ also points into this half-plane. Thus, $\mathbf{n}_e \cdot \mathbf{n}_{e''} > 0$.

Case (b): Since $\overline{pp'}$ is not contained in $\mathcal{P}$, the ray r must leave $\mathcal{P}$ at some point p'' on an edge $e'' \neq e$ (see Figure 5(iii)); again, note that $p' \equiv p''$, possibly. As in case (a), we have $\mathbf{n}_e \cdot \mathbf{n}_{e''} > 0$.

Thus we have shown that, for any edge e, if $\mathcal{P}$ is not a terrain with base e, then there is some edge $e'' \neq e$ such that $\mathbf{n}_e \cdot \mathbf{n}_{e''} > 0$. This completes the proof. ∎

2.1 The algorithm

Let $\mathcal{S}^1$ be the unit-circle of directions centered at the origin. Each edge normal $\mathbf{n}_e$ yields a *normal-point* n_e on $\mathcal{S}^1$. (Equal normals are represented by different normal-points.) Let N denote the set of normal-points on $\mathcal{S}^1$. By *the angle between two normal-points on* $\mathcal{S}^1$ we mean the angle subtended at the origin by the shorter of the two arcs of $\mathcal{S}^1$ that join the points. From Lemma 2.1 it follows that e is a base of $\mathcal{P}$ if and only if the angle between n_e and $n_{e'}$ is at least $\pi/2$, for all $e' \neq e$. This immediately gives an $O(n^2)$-time algorithm to test if $\mathcal{P}$ is a terrain.

The problem can be solved in $O(n \log n)$ time as follows. Pre-sort the normal-points around $\mathcal{S}^1$ (breaking ties arbitrarily) and store them in a doubly-linked circular list. Pick any point n_e in this list and check if the angle between n_e and its successor, and between n_e and its predecessor, are both at least $\pi/2$. If so, then the angle between n_e and any other point in the list is also at least $\pi/2$, so e is a base. Otherwise, repeat the above with the successor of n_e, and so on. The running time is dominated by the circular sort. (Some minor optimization is possible. If the predecessor and/or successor of n_e fails the test, it can be eliminated from further consideration since the test at this point will fail due to n_e.)

It is possible to improve the running time to $O(n)$ using the following lemma:

Lemma 2.2 *W.l.o.g. assume that no normal-point of N coincides with $(-1, 0)$ or $(1, 0)$; otherwise, rotate N suitably. Let n_a (resp. n_b) be the first (resp. last) clockwise point of N on the upper half of $\mathcal{S}^1$. Let n_c (resp. n_d) be the first (resp. last) clockwise point of N on the lower half of $\mathcal{S}^1$ (ties are broken arbitrarily). Then $\mathcal{P}$ is a terrain if and only if at least one of the edges a, b, c, or d is a base.*

Proof

($\Leftarrow$) Obvious.

($\Rightarrow$) Let $\mathcal{P}$ be a terrain and assume that none of a, b, c, or d is a base. Let e be any other edge of $\mathcal{P}$. Then n_e lies between n_a and n_b or between n_c and n_d. (Because of our choice of n_a, n_b, n_c, and n_d as extreme points, n_e cannot be between n_d and n_a or between n_b and n_c.) W.l.o.g. let n_e be between n_a and n_b. Since the angle between n_a and n_b is less than π, the angle between at least one of n_e and n_a or n_e and n_b is less than $\pi/2$. Therefore, e is not a base. Thus, no edge of $\mathcal{P}$ is a base and so $\mathcal{P}$ is not a terrain—a contradiction.

∎

By Lemma 2.2 it suffices to check just the points n_a, n_b, n_c, and n_d. These points can be identified in $O(n)$ time. For each point we check if each of the remaining points of N makes an angle of at least $\pi/2$, which takes time $O(n)$.

Theorem 2.1 *It is possible to decide in $O(n)$ time and $O(n)$ space whether an n-vertex simple polygon is a terrain. If it is a terrain, then a base can be identified within this time bound.*

Lemma 2.2 implies that a simple n-vertex polygon has at most four bases. Lemma 2.3 gives an alternative proof (in fact, the proof gives the slightly stronger result that $\mathcal{P}$ has at most three bases, if $n > 4$).

Lemma 2.3 *A simple polygon $\mathcal{P}$ with $n > 4$ vertices has at most three bases.*

Proof Suppose for a contradiction that $\mathcal{P}$ has more than three bases. Consider the subset N' of N consisting of the normal-points corresponding to four of these bases together with one additional normal-point (notice that N' exists, since $|N| > 4$). By Lemma 2.1, the distance on $\mathcal{S}^1$ between every consecutive pair of points from N' is at least $\pi/2$. Since there are five consecutive pairs of points in N', it follows that the circumference of $\mathcal{S}^1$ has length at least $5\pi/2$—a contradiction. ∎

The following lemma and corollary will be used in Section 3.

Lemma 2.4 *Let $\mathcal{P}$ be a simple polygon and let $\mathcal{Q}$ be its convex hull. If $\mathcal{P}(e)$ is a terrain, then e is an edge of $\mathcal{Q}$ and $\mathcal{Q}(e)$ is a terrain.*

Proof Let l be the line through e and let l^+ and l^- be the two open half-planes determined by l. We will show that all the vertices of $\mathcal{P}$, excluding the endpoints of e, must lie in either l^+ or l^-. This implies that e is an edge of $\mathcal{Q}$ [dBvKOS97, page 3].

Let V_e be the strip erected perpendicularly to e. Let $V_e^+ = l^+ \cap V_e$ and $V_e^- = l^- \cap V_e$. Since no point outside V_e can be connected to e by a perpendicular line segment, the polygon $\mathcal{P}$ is completely contained in V_e. Assume for a contradiction that there are vertices u and u' of $\mathcal{P}$ such that $u \in V_e^+$ and $u' \in V_e^-$. Note that u and u' cannot be the endpoints of e, since V_e^+ and V_e^- do not contain e. Consider the part of the boundary of $\mathcal{P}$, from u to u', which does not contain e. If we walk from u to u' along this part of the boundary, then we cross l but not e; hence we cross one of the bounding lines of V_e, a contradiction. Therefore, all the points of $\mathcal{P}$ must lie in either V_e^+ or V_e^-, which implies that e is an edge of $\mathcal{Q}$.

To show that $\mathcal{Q}(e)$ is a terrain, pick a point p' in $\mathcal{Q}$. If $p' \in \mathcal{P}$, then since $\mathcal{P}(e)$ is a terrain there is a point $p \in e$ such that $\overline{pp'}$ lies in $\mathcal{P}$ and is perpendicular to e. Any segment which lies in $\mathcal{P}$ must also lie in $\mathcal{Q}$ (by definition) and therefore $\overline{pp'}$ is completely contained in $\mathcal{Q}$.

If $p' \notin \mathcal{P}$, then let p'' be the first intersection between $\mathcal{P}$ and the ray from p' in direction $\mathbf{n}_e$. Since $\mathcal{P}(e)$ is a terrain, there is a point $p \in e$ such that $\overline{pp''}$ lies in $\mathcal{P}$ and is perpendicular to e. But then $\overline{pp'}$ is also perpendicular to e and lies completely in $\mathcal{Q}$ (by definition), which completes the proof that $\mathcal{Q}(e)$ is a terrain. ∎

Corollary 2.1 *If $\mathcal{P}(e)$ is a terrain, then the interior angles at e are at most $\pi/2$.*

Proof Follows from the fact that $\mathcal{P}$ lies in either V_e^+ or V_e^-, as shown in the proof of Lemma 2.4 ∎

3 Decomposing a simple polygon into two terrains with a common base

In what follows, we assume that in a pre-processing step, successive collinear edges of $\mathcal{P}$ have been merged into a single edge. Let l be a line whose intersection with $\mathcal{P}$ is a connected segment $\overline{uv}$. Let C^+ (resp. C^-) be the boundary of $\mathcal{P}$ from u to v (resp. v to u) taken in counter-clockwise order. Let $\mathcal{P}^+$ (resp. $\mathcal{P}^-$) be the polygon bounded by $C^+ \cup \overline{uv}$ (resp. $C^- \cup \overline{uv}$).

Lemma 3.1 *Let l be a line which decomposes $\mathcal{P}$ into two simple polygons $\mathcal{P}^+$ and $\mathcal{P}^-$, as above, such that $\mathcal{P}^+(\overline{uv})$ and $\mathcal{P}^-(\overline{uv})$ are both terrains. Then*

(i) u and v are both vertices of $\mathcal{P}$, or

(ii) u is a vertex of $\mathcal{P}$ and v is the point where l intersects perpendicularly the interior of an edge, e, of $\mathcal{P}$, or

(iii) u and v are points where l intersects perpendicularly the interiors of edges e_u and e_v of $\mathcal{P}$, respectively.

Proof If u and v are both vertices of $\mathcal{P}$, then case (i) holds. If u is a vertex and v is in the interior of an edge e, then by Corollary 2.1, the interior angle at v in $\mathcal{P}^+$ is at most $\pi/2$. Similarly, the interior angle at v in $\mathcal{P}^-$ is also at most $\pi/2$. Since the sum of these angles is π (because e is contained in a straight line), it follows that the two interior angles at v are both $\pi/2$. Thus, case (ii) holds. Finally, if both u and v are in the interior of edges of $\mathcal{P}$, then by a similar argument the two interior angles at u and at v are each $\pi/2$, so case (iii) holds. ∎

Lemma 3.2 *If $\mathcal{P}^+(\overline{uv})$ and $\mathcal{P}^-(\overline{uv})$ are both terrains, then the element-pair (u, v) or (u, e) or (e_u, e_v) in Lemma 3.1 determining the decomposing line l is an antipodal pair of the convex hull, $\mathcal{Q}$, of $\mathcal{P}$.*

Proof Let (x, y) be the element-pair determining the decomposing line. From the proof of Lemma 2.4 it follows that $\mathcal{P}$ is contained in the strip $V_{\overline{uv}}$ erected perpendicularly to $\overline{uv}$. Since x is either a vertex or an edge of $\mathcal{P}$ perpendicular to the line l containing $\overline{uv}$, it is clear that x is on a bounding line of $V_{\overline{uv}}$. Since $\mathcal{P}$ is completely on one side of this bounding line, x belongs to $\mathcal{Q}$ [dBvKOS97, page 3]. Similarly, y is on the other bounding line of $V_{\overline{uv}}$ and hence belongs to $\mathcal{Q}$. Since $\mathcal{P}$ is contained between the parallel bounding lines of $V_{\overline{uv}}$, it follows that (x, y) is an antipodal pair of $\mathcal{Q}$. ∎

By Lemma 3.2, it is sufficient to consider only the antipodal pairs of $\mathcal{Q}$ in computing the decomposing line l. There are three types of antipodal pairs to consider: $(vertex, vertex)$-pair, $(vertex, edge)$-pair, and $(edge, edge)$-pair, which we denote as VV-pair, VE-pair, and EE-pair, respectively. We discuss how to handle these in Sections 3.1–3.3, respectively. We assume for now that no two edges of $\mathcal{Q}$ are collinear—this case is considered in Section 3.4.

The results from Sections 3.1–3.4 are summarized in the following theorem:

Theorem 3.1 *It is possible to determine in $O(n \log n)$ time and $O(n)$ space whether a simple polygon $\mathcal{P}$ can be decomposed by a line l into two terrain polygons, $\mathcal{P}^+(e)$ and $\mathcal{P}^-(e)$, where $e = l \cap \mathcal{P}$. If l exists, it can be computed within the same bounds.*

3.1 Handling antipodal VV-pairs

Recall that $\mathcal{Q}$ is the convex hull of $\mathcal{P}$. Let $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_m$ be the vertices of $\mathcal{P}$ and $\mathcal{Q}$, respectively, in counter-clockwise order. (Each vertex q_i of $\mathcal{Q}$ corresponds to some vertex p_j of $\mathcal{P}$.)

Consider the set, L, of all lines, l_{ij}, defined by the antipodal VV-pairs (q_i, q_j). For each l_{ij} we can test whether it decomposes $\mathcal{P}$ into two simple polygons, $\mathcal{P}_{ij}^+$ and $\mathcal{P}_{ij}^-$, in $O(n)$ time. In

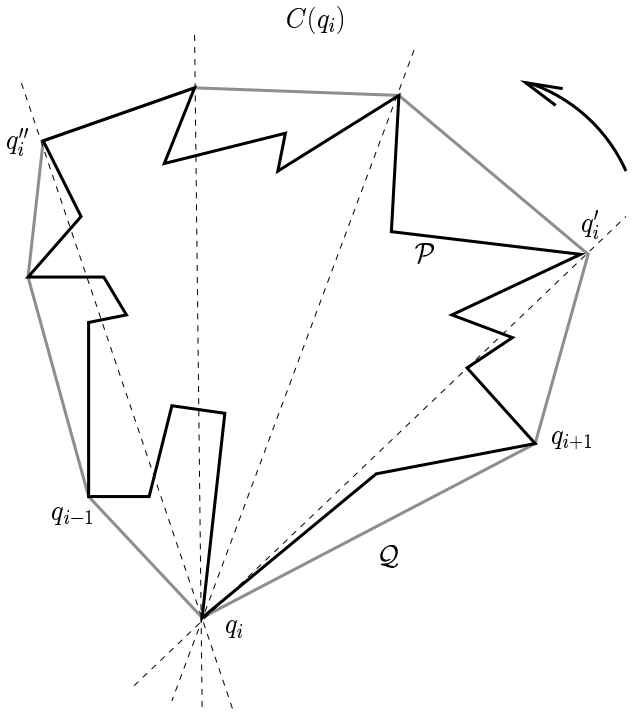

Figure 6: *Illustrating the algorithm for antipodal VV-pairs.*

---

additional $O(n)$ time we can test whether $\mathcal{P}_{ij}^{+}(e_{ij})$ and $\mathcal{P}_{ij}^{-}(e_{ij})$ are terrains, where $e_{ij} = l_{ij} \cap \mathcal{P}$, using Lemma 2.1. Since the number of antipodal VV-pairs is $O(n)$ the overall running time will be $O(n^2)$.

The run-time can be improved to $O(n \log n)$ by choosing carefully the order in which the lines in L are examined.

Let q_i be a vertex of $\mathcal{Q}$. The vertices antipodal to q_i form a chain, $C(q_i)$, along the boundary of $\mathcal{Q}$. Furthermore, if q_i' and q_i'' are the first and last vertices of $C(q_i)$ in counter-clockwise order, then q_i' is the last vertex of $C(q_{i-1})$, and q_i'' is the first vertex of $C(q_{i+1})$ [PS85, page 173]. Assuming for the moment that $\mathcal{Q}$ has no parallel edges, $C(q_{i+1})$ follows immediately after $C(q_i)$ and the two chains share the vertex q_i''. (See Figure 6.)

The antipodal pairs of $\mathcal{Q}$ can be enumerated using the *rotating calipers approach* [Tou83, HT85, PS85]: Let $u \equiv q_i$ and $v \equiv q_i'$. Advance v along $C(q_i)$ until v coincides with q_i''; each pair (u, v) is an antipodal pair. When v coincides with q_i'', u is advanced to q_{i+1} to reflect the fact that v is now at the first vertex of $C(q_{i+1})$, and the process is repeated. (However, if $\overline{q_i q_{i+1}}$ is parallel to $\overline{q_j q_{j+1}}$, then $C(q_i)$ and $C(q_{i+1})$ share $\overline{q_j q_{j+1}} = \overline{q_{i+1}' q_i''}$. In this case, v visits q_{i+1}' and q_i'' twice, i.e. v backtracks as u is advanced from q_i to q_{i+1}). The enumeration terminates when u and v reach the initial configuration. During this process, u and v visit each vertex of $\mathcal{Q}$ at most twice, so the enumeration takes $O(n)$ time.

The decomposition lines determined by antipodal VV-pairs can be computed during the above enumeration, as follows: We use data structures that can answer the following types of query given a polygon $\mathcal{P}$: (1) determine whether a line, l, through two vertices of $\mathcal{P}$ decomposes $\mathcal{P}$ into two

simple polygons; and (2) determine whether $\mathcal{P}(e)$ is a terrain, where e is an edge of $\mathcal{P}$.

For the first type of query we use the ray-shooting data structure of [HS95]. If the ray directed from u to v meets the boundary of $\mathcal{P}$ at v, then l_{uv} decomposes $\mathcal{P}$ into two simple polygons. (Notice that u and v are vertices of $\mathcal{P}$, since they are vertices of $\mathcal{Q}$.)

The second type of query can be answered efficiently using a leaf-oriented, balanced, binary-search tree, $\mathcal{T}$, whose leaves store the normal-points corresponding to the edges of $\mathcal{P}$ in the order in which they appear around the unit-circle $\mathcal{S}^1$. Given an edge e of $\mathcal{P}$, by Lemma 2.1 it is sufficient to locate the neighbors of n_e in $\mathcal{T}$ and check whether they are each at distance at least $\pi/2$ from n_e. The time to build a tree for n points is $O(n \log n)$ and the query time per point is $O(\log n)$.

We use separate trees for $\mathcal{P}^+$ and $\mathcal{P}^-$, denoted $\mathcal{T}^+$ and $\mathcal{T}^-$, respectively, which maintain the invariant that whenever a line l from L decomposes $\mathcal{P}$ into two simple polygons, $\mathcal{T}^+$ and $\mathcal{T}^-$ contain the normal-points corresponding to the edges of $\mathcal{P}^+$ and $\mathcal{P}^-$, respectively, except for $e = l \cap \mathcal{P}$. What remains to be shown is how to maintain $\mathcal{T}^+$ and $\mathcal{T}^-$ efficiently as we consider different antipodal VV-pairs (u, v).

Let $\mathcal{Q}^+$ be the polygon defined by the chain of edges along the boundary of $\mathcal{Q}$ in counter-clockwise order from u to v, together with the edge $\overline{uv}$. Define $\mathcal{Q}^-$ similarly. Since $\mathcal{Q}$ is a convex polygon, the line l_{uv} containing $\overline{uv}$ will always decompose $\mathcal{Q}$ into two simple polygons. Whenever l_{uv} also decomposes $\mathcal{P}$ into two simple polygons, we let $\mathcal{P}^+$ be the polygon defined by the chain of edges along the boundary of $\mathcal{P}$ in counter-clockwise order from u to v, together with the edge $\overline{uv}$. $\mathcal{P}^-$ is defined similarly.

Let $\overline{q_k q_{k+1}}$ be an edge of $\mathcal{Q}$ and let $H(q_k, q_{k+1})$ denote the chain of edges along the boundary of $\mathcal{P}$ from q_k to q_{k+1} in counter-clockwise order. Clearly, if a line l from L decomposes $\mathcal{P}$ into two simple polygons, then $\overline{q_k q_{k+1}}$ is an edge of $\mathcal{Q}^+$ if and only if $H(q_k, q_{k+1})$ belongs to $\mathcal{P}^+$. Therefore, it is sufficient to keep track of the edges of $H(q_k, q_{k+1})$ only when $\overline{q_k q_{k+1}}$ changes from $\mathcal{Q}^+$ to $\mathcal{Q}^-$ and vice versa. This offers an efficient way to maintain $\mathcal{T}^+$ and $\mathcal{T}^-$. Each edge $\overline{q_k q_{k+1}}$ of $\mathcal{Q}$ changes exactly twice between $\mathcal{Q}^+$ and $\mathcal{Q}^-$, and this happens precisely when either u or v is advanced from q_k to q_{k+1}. Whenever $\overline{q_k q_{k+1}}$ changes from $\mathcal{Q}^+$ to $\mathcal{Q}^-$, the normal-points corresponding to the edges of $H(q_k, q_{k+1})$ are deleted from $\mathcal{T}^+$ and inserted in $\mathcal{T}^-$. Since, for any two edges, $\overline{q_k q_{k+1}}$ and $\overline{q_j q_{j+1}}$, $H(q_k, q_{k+1})$ and $H(q_j, q_{j+1})$ have no edges in common, each edge of $\mathcal{P}$ will be inserted and deleted exactly twice from $\mathcal{T}^+$ and $\mathcal{T}^-$. (If $\overline{q_i q_{i+1}}$ is parallel to $\overline{q_k q_{k+1}}$, then $C(q_i)$ and $C(q_{i+1})$ share the edge $\overline{q_k q_{k+1}}$, so $\overline{q_k q_{k+1}}$ will switch twice more between $\mathcal{Q}^+$ and $\mathcal{Q}^-$, as v backtracks from q_{k+1} to q_k. However, this does not affect the overall running time.)

Throughout the algorithm, the size of $\mathcal{T}^+$ and $\mathcal{T}^-$ is $O(n)$. The time to build $\mathcal{T}^+$ and $\mathcal{T}^-$ is $O(n \log n)$, and each insert and delete operation takes time $O(\log n)$. Since each edge of $\mathcal{P}$ is inserted and deleted $O(1)$ times, the overall time for the updates is $O(n \log n)$. The time to query $\mathcal{T}^+$ and $\mathcal{T}^-$ for the terrain property of $\mathcal{P}^+$ and $\mathcal{P}^-$, respectively, is $O(\log n)$, or an overall time of $O(n \log n)$ since there are $O(n)$ antipodal VV-pairs.

The pre-processing time to build the ray-shooting data structure is $O(n)$ and its space requirement is $O(n)$. Each ray shooting query takes $O(\log n)$ time, or $O(n \log n)$ time over all antipodal VV-pairs.

3.2 Handling antipodal VE-pairs

The antipodal VE-pairs can be handled simultaneously with the antipodal VV-pairs by a simple extension of the algorithm of Section 3.1. We identify the antipodal VE-pairs as follows: Let u

be a vertex of $\mathcal{Q}$. The endpoints of the edges antipodal to u must be in $C(u)$. Let $W(u)$ be the double-wedge defined by the lines through the edges incident to u and not containing $\mathcal{Q}$. Then, for any edge e whose endpoints are in $C(u)$, (u, e) is an antipodal VE-pair if and only if the line through u that is parallel to the line through e is in $W(u)$. Thus, the edges antipodal to u can be computed while enumerating the vertices antipodal to u. Clearly, the total number of antipodal VE-pairs is $O(n)$.

Our goal now is to identify all lines, l, determined by VE-pairs, (u, e), of $\mathcal{P}$ that satisfy case (ii) of Lemma 3.1. By Lemma 3.2, it follows that (u, e) is an antipodal VE-pair of $\mathcal{Q}$, and e is an edge of both $\mathcal{P}$ and $\mathcal{Q}$.

Let $\overline{q_i q_{i+1}}$ be an edge of $\mathcal{P}$ and $\mathcal{Q}$ antipodal to u. We can handle $(u, \overline{q_i q_{i+1}})$ during the transition from (u, q_i) to (u, q_{i+1}) in the algorithm of Section 3.1. We first check whether the line through u and perpendicular to the line through $\overline{q_i q_{i+1}}$ intersects $\overline{q_i q_{i+1}}$ and decomposes $\mathcal{P}$ into two simple polygons. This is the case if the ray from u in the direction of the outer normal of $\overline{q_i q_{i+1}}$ meets the boundary of $\mathcal{P}$ at a point in the interior of $\overline{q_i q_{i+1}}$. We then check whether the two simple polygons are terrains.

Recall that for (u, q_i) we have associated data structures $\mathcal{T}^+$ and $\mathcal{T}^-$. When transitioning to $(u, \overline{q_i q_{i+1}})$, $\mathcal{P}^+$ acquires a portion of $\overline{q_i q_{i+1}}$. Therefore, we insert the normal-point for $\overline{q_i q_{i+1}}$ in $\mathcal{T}^+$ (note that $\mathcal{T}^-$ does not need to be updated). We then test if $\mathcal{P}^+$ and $\mathcal{P}^-$ are terrains w.r.t. the edge $l \cap \mathcal{P}$.

When we next transition from $(u, \overline{q_i q_{i+1}})$ to (u, q_{i+1}) we update $\mathcal{T}^-$ by deleting from it the normal point for $\overline{q_i q_{i+1}}$ ($\mathcal{T}^+$ does not need to be updated).

As in Section 3.1, the time for these steps is $O(n \log n)$.

3.3 Handling antipodal EE-pairs

The antipodal EE-pairs present challenges not encountered in the previous two cases. While in the case of antipodal VV-pairs and VE-pairs the decomposition line is determined uniquely, in the case of antipodal EE-pairs only the direction of the decomposition line is known. However, it is still possible to determine efficiently a suitable decomposition line (if it exists), as we show below.

We would like to identify all lines, l, determined by EE-pairs, (e', e''), of $\mathcal{P}$ that satisfy case (iii) of Lemma 3.1. By Lemma 3.2, it follows that (e', e'') is an antipodal EE-pair of $\mathcal{Q}$, and e' and e'' are edges of both $\mathcal{P}$ and $\mathcal{Q}$.

The antipodal EE-pairs can be enumerated using the following observation: Let (u, v) be an antipodal VV-pair, let u^- be the clockwise neighbor of u, and let v^+ be the counter-clockwise neighbor of v. If both u and u^- are antipodal to $\overline{vv^+}$, then $(\overline{u^-u}, \overline{vv^+})$ is an antipodal EE-pair. Therefore, the antipodal EE-pairs can be enumerated while enumerating the antipodal VV-pairs.

The antipodal EE-pair $(\overline{u^-u}, \overline{vv^+})$ determines a family of parallel lines perpendicular to the lines through $\overline{u^-u}$ and $\overline{vv^+}$. Let $\mathbf{d}$ be the unit-normal that is directed from u^- to u. Let $H(u, v)$ and $H(v^+, u^-)$ be the counter-clockwise chains of vertices from u to v and v^+ to u^-, respectively. Let r be the extreme vertex of $H(u, v)$ in direction $-\mathbf{d}$ and let s be the extreme vertex of $H(v^+, u^-)$ in direction $\mathbf{d}$. Let l_r be the line through r that is perpendicular to $\mathbf{d}$. Clearly, the vertices of $H(u, v)$ are all in exactly one of the closed half-planes of l_r, which we denote as l_r^+. Then l_r decomposes $\mathcal{P}$ into two simple polygons if and only if s is not in l_r^+ (see Figure 7). If l_s is the line through s that is perpendicular to $\mathbf{d}$, then any line, l, in the open strip determined by l_r and l_s is a candidate decomposition line.

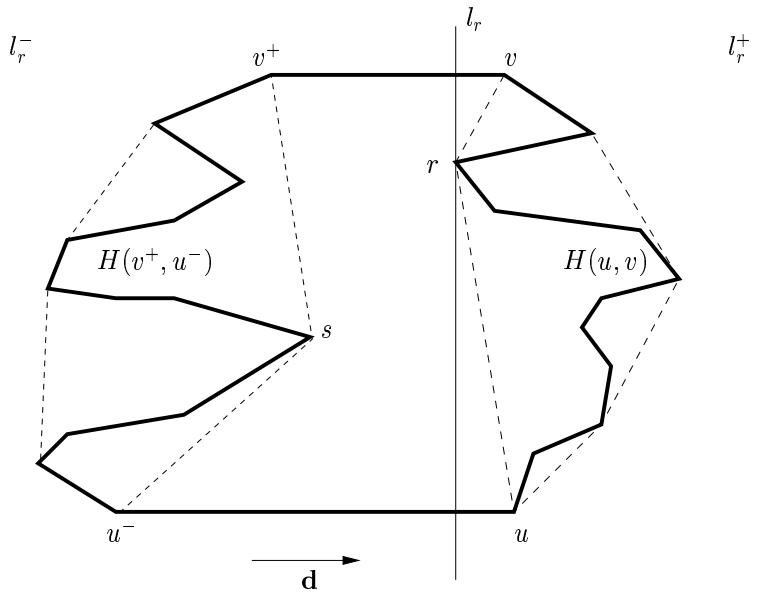

Figure 7: *Illustrating the algorithm for antipodal EE-pairs. The dashed polygons are the convex hulls of $H(u, v)$ and $H(v^+, u-)$.*

To find the extreme vertex of $H(u, v)$ in a given direction it is sufficient to consider the convex hull of $H(u, v)$. To answer an extreme-vertex query efficiently, we use a data structure for dynamic convex hull maintenance [OvL81, Ove81, FHS96, Cha01, BJ00]. This structure needs to be dynamic since $H(u, v)$ changes as we consider different antipodal EE-pairs. Since the changes occur only at the ends of a simple path we will use the data structure of [FHS96] to maintain the convex hull of $H(u, v)$. This data structure has amortized query and update time of $O(\log n)$ for a sequence of n operations. Denote this structure by $\mathcal{H}^+$. Similarly, let $\mathcal{H}^-$ be the data structure for maintaining the convex hull of $H(v^+, u^-)$.

Suppose that l decomposes $\mathcal{P}$ into two simple polygons, $\mathcal{P}^+$ and $\mathcal{P}^-$, as determined by querying $\mathcal{H}^+$ and $\mathcal{H}^-$. We need to test whether $\mathcal{P}^+(e_r)$ and $\mathcal{P}^-(e_r)$ are terrains, where $e_r = l_r \cap \mathcal{P}$. We update $\mathcal{T}^+$ and $\mathcal{T}^-$ to reflect the fact that portions of $\overline{u^-u}$ and $\overline{vv^+}$ are in $\mathcal{P}^+$ and $\mathcal{P}^-$ by inserting the corresponding normal-points. We query $\mathcal{T}^+$ with the normal-point corresponding to $-\mathbf{d}$ to determine if $\mathcal{P}^+(e_r)$ is a terrain. Similarly, we query $\mathcal{T}^-$ with the normal-point for $\mathbf{d}$ to determine if $\mathcal{P}^-(e_r)$ is a terrain.

The overall run-time is dominated by the time to maintain $\mathcal{H}^+$ and $\mathcal{H}^-$ whenever the chains $H(u, v)$ and $H(v^+, u^-)$ change. These chains change once when u or v is advanced in the handling of antipodal VV-pairs (see Section 3.1). Since there are $O(n)$ antipodal VV-pairs the overall time is $O(n \log n)$. Each extreme-vertex query on $\mathcal{H}^+$ and $\mathcal{H}^-$ takes $O(\log n)$ time or $O(n \log n)$ for all antipodal EE-pairs.

The total time to update $\mathcal{T}^+$ and $\mathcal{T}^-$ is $O(n \log n)$ since for each antipodal EE-pair two points are inserted and deleted. Additionally, $O(\log n)$ time is spent for testing whether $\mathcal{P}^+$ and $\mathcal{P}^-$ are terrains, or $O(n \log n)$ over all antipodal EE-pairs.

3.4 Handling collinear edges

The run-time analysis of the algorithms in Sections 3.1–3.3 relies on the assumption that the number of antipodal pairs of $\mathcal{Q}$ is $O(n)$. In the presence of collinear edges this assumption is violated since the number of antipodal pairs could, in fact, be $\Theta(n^2)$. (Consider for instance a convex polygon containing two parallel chains of edges, each of size $\Theta(n)$. Every pair of edges, one from each chain, constitutes an antipodal pair.) In this section, we establish a sufficient condition for a line, l, to not be a valid decomposition line, and then show that this condition holds for all lines determined by antipodal pairs of collinear edges and/or vertices of $\mathcal{Q}$. Thus, these antipodal pairs can be ignored, thereby keeping the number of antipodal pairs of interest within $O(n)$.

Lemma 3.3 *Let l be a line which decomposes $\mathcal{P}$ into two simple polygons, $\mathcal{P}^+$ and $\mathcal{P}^-$, and let $e_l = \mathcal{P} \cap l$. If there is a line l' perpendicular to l, which contains a pair of vertices (q_k, q_{k+1}) such that $\overline{q_k q_{k+1}}$ is an edge of $\mathcal{Q}$, but not of $\mathcal{P}$, then at least one of $\mathcal{P}^+$ and $\mathcal{P}^-$ is not a terrain w.r.t. e_l.*

Proof We claim that at least one of q_k and q_{k+1} cannot be connected to e_l by a perpendicular line segment without violating the terrain property for $\mathcal{P}^+$ or $\mathcal{P}^-$ w.r.t. e_l.

Suppose that l does not intersect the interior of $\overline{q_k q_{k+1}}$ and w.l.o.g. assume that $\overline{q_k q_{k+1}}$ is in $\mathcal{Q}^+$; then q_k and q_{k+1} are vertices of $\mathcal{P}^+$. Let s_k and s_{k+1} be segments contained in l' that connect q_k and q_{k+1}, respectively, to the point $p = l \cap l'$. Note that s_k and s_{k+1} are perpendicular to l. Furthermore, one of s_k or s_{k+1} contains $\overline{q_k q_{k+1}}$ completely; w.l.o.g assume that $\overline{q_k q_{k+1}}$ is contained in s_k. If $p \notin e_l$, then the claim clearly holds. Otherwise, since $\overline{q_k q_{k+1}}$ is an edge of $\mathcal{Q}$, but not of $\mathcal{P}$, $\overline{q_k q_{k+1}}$ is in the exterior of $\mathcal{P}$. Therefore, s_k is not contained entirely in $\mathcal{P}^+$, and the claim follows.

Now, suppose that l intersects the interior of $\overline{q_k q_{k+1}}$ and w.l.o.g. assume that q_k and q_{k+1} are vertices of $\mathcal{Q}^+$ and $\mathcal{Q}^-$, respectively; then q_k and q_{k+1} are vertices of $\mathcal{P}^+$ and $\mathcal{P}^-$, respectively. Again, s_k and s_{k+1} are contained in l' and share a common endpoint, and, furthermore, s_k and s_{k+1} are both contained in $\overline{q_k q_{k+1}}$. Therefore, s_k and s_{k+1} are in the exterior of $\mathcal{P}$, and the claim follows. ∎

Now, consider the following antipodal pairs determining lines, l, that decompose $\mathcal{P}$ into two simple polygons:

(i) *antipodal EE-pair, (e, e'), where e and e' are edges of $\mathcal{P}$ and $\mathcal{Q}$*
By Lemma 3.1 the decomposition line, l, determined by the antipodal EE-pair, (e, e'), is perpendicular to e', and therefore, is perpendicular to the line, l', through e'. Suppose there is an edge of $\mathcal{Q}$, which is collinear with e'. Then by convexity of $\mathcal{Q}$, there must be an edge e'', which is also collinear with e' and is adjacent to e' on the boundary of $\mathcal{Q}$. Moreover, e'' is not an edge of $\mathcal{P}$, since $\mathcal{P}$ does not have collinear edges (by assumption). Thus, by Lemma 3.3, l does not decompose $\mathcal{P}$ into two terrains.

(ii) *antipodal VE-pair, (v, e'), where v is a vertex of $\mathcal{P}$ and e' is an edge of $\mathcal{P}$ and $\mathcal{Q}$*
Similar to the previous case, the line, l, determined by the antipodal VE-pair, (v, e'), does not decompose $\mathcal{P}$ into two terrains if there is an edge e'' of $\mathcal{Q}$ which is collinear with e'.

(iii) *antipodal VV-pair, (v, v'), where v and v' are vertices of $\mathcal{P}$ and $\mathcal{Q}$ and v' is incident to two collinear edges of $\mathcal{Q}$*
We claim that the line l determined by the antipodal VV-pair (v, v') does not decompose $\mathcal{P}$

into two terrains. Assume for a contradiction that l decomposes $\mathcal{P}$ into two terrains, $\mathcal{P}^+(e_l)$ and $\mathcal{P}^-(e_l)$, where $e_l = l \cap \mathcal{P}$. Let e' and e'' be consecutive collinear edges of $\mathcal{Q}$ and let v' be the vertex incident to e' and e''. By Lemma 2.4, $\mathcal{Q}^+(e_l)$ and $\mathcal{Q}^-(e_l)$ are also terrains; w.l.o.g. assume that e' and e'' are edges of $\mathcal{Q}^+$ and $\mathcal{Q}^-$, respectively. By Corollary 2.1, the interior angles at v' in $\mathcal{Q}^+$ and $\mathcal{Q}^-$ are no greater than $\pi/2$. Furthermore, the interior angles at v' must sum to π since they are determined by collinear edges of $\mathcal{Q}$. Therefore, the interior angles at v' are exactly $\pi/2$, or equivalently, l is perpendicular to l'. Since e' and e'' are contained in l' and one of e' and e'' is an edge of $\mathcal{Q}$, but not of $\mathcal{P}$, by Lemma 3.3, it follows that l does not decompose $\mathcal{P}$ into two terrains, which is a contradiction; therefore, the claim holds.

The above discussion shows that collinear edges and/or vertices of $\mathcal{Q}$ cannot participate in antipodal pairs that determine lines, which decompose $\mathcal{P}$ into two terrains. Therefore, collinear edges can be merged into single edges during the construction of $\mathcal{Q}$. Recall that the algorithms from Sections 3.2–3.3 consider only the edges that are in both $\mathcal{P}$ and $\mathcal{Q}$, and therefore, will correctly ignore the edges that have resulted from a merge. Also, as a result of the merge, vertices incident to collinear edges will be eliminated correctly.

4 Decomposing a simple polygon into two terrains

In this section, we consider the general problem of decomposing $\mathcal{P}$ into two terrains that do not necessarily have a common base. Lemma 4.1 below characterizes the decomposition line.

We first define the notion of a *cusp*. A vertex v of $\mathcal{P}$ is a *cusp* w.r.t. a line l containing v if both of v's neighbors in $\mathcal{P}$ are in the same closed half-plane of l and the interior angle at v is strictly greater than π. An edge e of $\mathcal{P}$ is a *cusp* w.r.t. a line l containing e if both of e's endpoints are cusps. Note that if a line l contains a vertex or an edge of $\mathcal{P}$ that is a cusp w.r.t. l, then l decomposes $\mathcal{P}$ into more than two simple polygons.

Lemma 4.1 *Let L be a non-empty family of lines that decompose $\mathcal{P}$ into two non-empty terrains that do not have a common base. Then there is a line, l, in L which intersects the boundary of $\mathcal{P}$ at points u and v, such that*

(i) u and v are vertices of $\mathcal{P}$ and neither u nor v is a cusp w.r.t. l, or

(ii) the segment joining u and v contains an edge of $\mathcal{P}$ that is not a cusp w.r.t. l, or

(iii) u is a vertex of $\mathcal{P}$ that is not a cusp w.r.t. l, v is in the interior of an edge of $\mathcal{P}$, and l is perpendicular to the line containing some edge of $\mathcal{P}$

Proof Let l be a line in L. Since l decomposes $\mathcal{P}$ into two non-empty simple polygons, $l \cap \mathcal{P}$ contains exactly one line segment. We take u and v to be the endpoints of this segment and denote the segment by $\overline{uv}$. If u and v are vertices of $\mathcal{P}$, then neither can be a cusp with respect to l, since otherwise l would decompose $\mathcal{P}$ into more than two simple polygons. Similarly, if $\overline{uv}$ contains an edge of $\mathcal{P}$, that edge cannot be a cusp. Therefore, case (i) or case (ii) holds.

If u and v are not vertices of $\mathcal{P}$, let $\mathcal{P}^+_{\overline{uv}}$ be the terrain for which $\overline{uv}$ is not a base. Let r be the vertex of $\mathcal{P}^+_{\overline{uv}}$ that is closest to l, and let l_r be the line l translated to r. We first show that r is not

a cusp with respect to l_r. Assume for a contradiction that r is a cusp. Let e' and e'' be the edges of $\mathcal{P}^+_{\overline{uv}}$ incident to r and let α' and α'' be the portions of the interior angle at r between l_r and each of e' and e'', respectively (note that $\alpha' + \alpha'' < \pi$). Consider the outward-directed unit-normals of $\overline{uv}$, e', and e'' translated to r. The normal-points $n_{\overline{uv}}$, $n_{e'}$, and $n_{e''}$, partition the unit-circle into three disjoint arcs of lengths $\alpha' + \alpha''$, $\pi - \alpha'$, and $\pi - \alpha''$, respectively. Since each arc is of length less than π, this implies that no point on the unit-circle is at distance at least $\pi/2$ from each of $n_{\overline{uv}}$, $n_{e'}$, and $n_{e''}$. However, $\mathcal{P}^+_{\overline{uv}}(e)$ is a terrain, for some edge e of $\mathcal{P}^+_{\overline{uv}}$, and, furthermore, e is distinct from $\overline{uv}$, e', and e'' (the interior angle at r is greater than π, and, therefore, neither e' nor e'' can be a base for $\mathcal{P}^+_{\overline{uv}}$). Therefore, n_e must be at distance at least $\pi/2$ from each of $n_{\overline{uv}}$, $n_{e'}$, and $n_{e''}$. Thus, we have established a contradiction, and, therefore, r is not an interior cusp with respect to l_r. (This proof can also be adapted to show that l_r does not go through an edge of $\mathcal{P}$ which is a cusp; in this case we take e' and e'' to be the edges adjacent to the alleged cusp and arrive at a contradiction).

Since r is not a cusp w.r.t. l_r and since no vertex is strictly crossed during the translation of l to r, it follows that l_r also decomposes $\mathcal{P}$ into two simple polygons. Furthermore, since the sets of normal-points associated with the two polygons in the decomposition remains the same during the translation, it follows that l_r also decomposes $\mathcal{P}$ into two terrains; thus $l_r \in L$.

If l_r goes through an edge of $\mathcal{P}$, then case (ii) holds. Otherwise, let $\overline{qr} = l_r \cap int(\mathcal{P})$ and w.l.o.g. let $\mathcal{P}^+_{\overline{qr}}$ be the terrain for which $\overline{qr}$ is not a base. Let l'_r be the line l_r rotated by an angle β about r in the half-plane containing $\mathcal{P}^+_{\overline{qr}}$, where β is the smallest angle such that l'_r either goes through a vertex r' of $\mathcal{P}$ or becomes perpendicular to the line containing an edge of $\mathcal{P}$; note that l'_r is in L. If the latter case applies, then case (iii) holds. Otherwise, similar to the earlier discussion, we can show that l'_r does not go through a vertex or an edge that is a cusp of $\mathcal{P}$ with respect to l'_r, and therefore, either case (i) or case (ii) holds. ∎

Using the characterization in Lemma 4.1, it is possible to decide whether $\mathcal{P}$ can be decomposed into two terrains, as follows: Enumerate all lines that go through two vertices of $\mathcal{P}$ (case (i) and (ii)), or go through a vertex of $\mathcal{P}$ and are perpendicular to the line containing an edge of $\mathcal{P}$ (case (iii)). For each line test whether it decomposes $\mathcal{P}$ into two polygons $\mathcal{P}^+$ and $\mathcal{P}^-$, and whether $\mathcal{P}^+$ and $\mathcal{P}^-$ are terrains. Clearly, the problem can be solved in time $O(n^3)$. In what follows we present an improved algorithm, whose performance is summarized in the following theorem.

Theorem 4.1 *It is possible to determine in $O(n^2 \log n)$ time and $O(n)$ space whether there is a line which decomposes a polygon $\mathcal{P}$ into two terrains. If such a line exists, it can be computed within the same bounds.*

4.1 The Algorithm

(a) *Handling the lines determined by two vertices of $\mathcal{P}$:*
For each vertex u of $\mathcal{P}$ we enumerate the lines through u by visiting the vertices of $\mathcal{P}$ from u^+ to u^- in counter-clockwise order. Let v be a vertex of $\mathcal{P}$ and let C^+ and C^- be the boundary of $\mathcal{P}$ from u to v, and v to u, respectively, taken in counter-clockwise order. During the walk from u^+ to u^- we maintain two leaf-oriented, balanced, binary-search trees, $\mathcal{T}^+$ and $\mathcal{T}^-$, whose leaves store the normal-points corresponding to the edges of $C^+ \cup \overline{uv}$ and $C^- \cup \overline{uv}$, respectively, in the order in which they appear around the unit-circle $\mathcal{S}^1$ (the edge $\overline{uv}$ appears with different normals in $\mathcal{T}^+$ and $\mathcal{T}^-$). Whenever the line l_{uv} through u and v decomposes

$\mathcal{P}$ into two simple polygons, $\mathcal{P}^+$ and $\mathcal{P}^-$, the normal-points corresponding to their edges are stored in $\mathcal{T}^+$ and $\mathcal{T}^-$, respectively.

Initially $\mathcal{T}^+$ contains only the normal-point corresponding to $\overline{uu^+}$, $\mathcal{T}^-$ contains the normal-points corresponding to the edges of $\mathcal{P}$, and v is the vertex u^+. During the transition from v to v^+ we perform the following steps:

First, we update $\mathcal{T}^+$ and $\mathcal{T}^-$, to reflect the fact that the edge $\overline{vv^+}$ switches from C^- to C^+, and that the edge $\overline{uv}$ is replaced by $\overline{uv^+}$. The normal-point corresponding to $\overline{vv^+}$ is removed from $\mathcal{T}^-$ and inserted in $\mathcal{T}^+$, and the normal-points corresponding to $\overline{uv}$ in $\mathcal{T}^+$ and $\mathcal{T}^-$ are replaced by the appropriate normal-points corresponding to $\overline{uv^+}$.

Next, we test whether l_{uv} decomposes $\mathcal{P}$ into two simple polygons. This is equivalent to testing whether the ray originating from a point at infinity along l_{uv} and directed towards u intersects the boundary of $\mathcal{P}$ exactly twice. To perform the test efficiently we use the ray-shooting data structure of [HS95].

Finally, if l_{uv} decomposes $\mathcal{P}$ into two simple polygons, $\mathcal{P}^+$ and $\mathcal{P}^-$, we test whether each polygon is a terrain. To test $\mathcal{P}^+$ we find in $\mathcal{T}^+$ the neighbors, n_a and n_d, of the normal-point $(-1, 0)$, and the neighbors, n_b and n_c, of the normal-point $(1, 0)$. As shown in the proof of Lemma 2.2, these are the only candidates for bases of $\mathcal{P}^+$. For each candidate normal-point it is sufficient to test whether its neighbors in $\mathcal{T}^+$ are at distance at least $\pi/2$, since $\mathcal{T}^+$ stores the normal points in a sorted, doubly-linked, circular list at the leaves. (If any of $(-1, 0)$ and $(1, 0)$ is in $\mathcal{T}^+$, they are also considered as candidate bases.) Similarly, we use $\mathcal{T}^-$ to test whether $\mathcal{P}^-$ is a terrain.

Analysis: The initialization step is dominated by the time to build $\mathcal{T}^-$ with $O(n)$ points, and therefore, it takes $O(n \log n)$ time. During the transition from v to v^+ we perform a constant number of $O(\log n)$-time update operations to maintain $\mathcal{T}^+$ and $\mathcal{T}^-$. The ray-shooting data structure has $O(\log n)$ query time per ray, so testing whether $\mathcal{P}$ is decomposed into two simple polygons takes additional $O(\log n)$ time. Each look-up operation on $\mathcal{T}^+$ and $\mathcal{T}^-$ takes $O(\log n)$, and finding the neighbors of a normal-point in a doubly-linked list takes constant time. Therefore, the time to test the candidate base points is $O(\log n)$. Taken over all lines through u the run-time is $O(n \log n)$, or $O(n^2 \log n)$ over all pairs of vertices of $\mathcal{P}$. (The overall run-time dominates the time to set up the ray-shooting data structure, which needs to be done only once and takes $O(n)$ time.)

Throughout the algorithm, the size of $\mathcal{T}^+$ and $\mathcal{T}^-$ is $O(n)$, since together they contain the normal-points corresponding to the edges of $\mathcal{P}$. The space requirement for the ray-shooting data structure is also $O(n)$.

(b) *Handling the lines that go through a vertex of $\mathcal{P}$ and are perpendicular to the line containing an edge of $\mathcal{P}$:*
Let e be any edge of $\mathcal{P}$ and let l_e be the line containing e. We sweep a line, l, perpendicular to l_e which visits the vertices of $\mathcal{P}$ in sorted order in a direction parallel to l. We define l^+ to be the closed half-plane in the direction of the sweep, and l^- be the closed half-plane in the opposite direction. During the sweep we maintain data structures $\mathcal{T}^+$ and $\mathcal{T}^-$ similar to the ones described in part (a). Here $\mathcal{T}^+$ and $\mathcal{T}^-$ contain the normal-points corresponding to the edges of $\mathcal{P}$ (or portions thereof) that are in l^+ and l^-, respectively, and the normal-points

corresponding to the directions normal to l and pointing into l^- and l^+, respectively. Whenever l decomposes $\mathcal{P}$ into two simple polygons, $\mathcal{P}^+$ and $\mathcal{P}^-$, the normal-points corresponding to their edges are stored in $\mathcal{T}^+$ and $\mathcal{T}^-$, respectively.

Initially $\mathcal{T}^-$ is empty and $\mathcal{T}^+$ contains the normal-points corresponding to the edges of $\mathcal{P}$. At each vertex we consider the incident edges. The normal-points of the edges that lie in l^- are removed from $\mathcal{T}^+$. Next, we test whether l decomposes $\mathcal{P}$ into two terrains as described in part (a). Finally, the normal-points of the edges that lie in l^+ are inserted in $\mathcal{T}^-$.

Analysis: During the initialization step, the time to sort the vertices and build $\mathcal{T}^+$ is $O(n \log n)$. At each event, we perform two update operations per vertex, and therefore, over all vertices the run-time is $O(n \log n)$. The time to check if l decomposes $\mathcal{P}$ into two simple polygons is $O(\log n)$ per event, as discussed in part (a), or $O(n \log n)$ for the whole sweep. Since the sweep is performed for each edge of $\mathcal{P}$ the overall run-time is $O(n^2 \log n)$. The space per sweep is $O(n)$ and this can be re-used.

Combining the analyses for parts (a) and (b) gives the result in Theorem 4.1

Remark 4.1 In the above discussion we used ray-shooting to test whether the line l_{uv} determined by vertices u and v of $\mathcal{P}$ decomposes $\mathcal{P}$ into two simple polygons. This was achieved by testing whether the ray from a point at infinity along l_{uv} intersects the boundary of $\mathcal{P}$ exactly twice. If there are vertices that are collinear with u and v, the ray will meet the boundary of $\mathcal{P}$ at each such vertex and, therefore, it must be propagated along l_{uv}. This can affect the overall run-time, since we have to perform this operation for any pair of vertices that determines a line which coincides with l_{uv}. However, it is sufficient to propagate the ray along l_{uv} just once, since each line that coincides with l_{uv} yields the same configuration. Therefore, we perform the test only when v is the first vertex along l_{uv} that meets the ray, i.e. v is the extreme vertex along l_{uv}.

The overall run time for a given vertex u remains $O(n \log n)$. For each vertex we test whether it is the extreme vertex in $O(\log n)$ time using ray-shooting. For each line through u we propagate the ray exactly once, so over all lines the ray shooting takes $O(n \log n)$ time. (Note that if the ray intersects properly the boundary of $\mathcal{P}$ more than twice we can abort the ray-shooting, since then the line decomposes $\mathcal{P}$ into more than two simple polygons.)

5 Conclusions and future work

In this paper, we have presented algorithms to decide whether a simple polygon can be decomposed by a line into two terrains. If such decompositions exist, the algorithms compute a decomposing line and a base for each terrain. The algorithms use $O(n)$ space and run in $O(n \log n)$ time, if the terrains have a common base, and $O(n^2 \log n)$ time, otherwise. The algorithms are based on the rotating-calipers and sweep-line approach, and use efficient data structures for performing ray-shooting queries and dynamic maintenance of the convex hull of a polygonal path.

The terrain decomposition problem has application in Layered Manufacturing. It can be used to determine whether long and slender 3D objects can be decomposed into two pieces that do not need supports during the build phase. Furthermore, since the height-to-width ratio is decreased, the part has better stability and the number of layers that need to be processed is reduced. In future work, we plan to investigate possible improvements in the $O(n^2 \log n)$ running time of the algorithm for the case where the terrains do not have a common base.

References

[ABB+97] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu. Feasibility of design in stereolithography. *Algorithmica*, 19:61–83, 1997.

[BBvK97] P. Bose, D. Bremner, and M. van Kreveld. Determining the castability of simple polyhedra. *Algorithmica*, 19(1–2):84–113, September 1997.

[BJ00] G. Brodal and R. Jacob. Dynamic planar convex hull with optimal query time and $O(\log n \cdot \log \log n)$ update time. In *Proc. 7th Scand. Workshop Algorithm Theory*, volume 1851 of *Lecture Notes Comput. Sci.*, pages 57–70. Springer-Verlag, 2000.

[Cha01] T. M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *J. ACM*, 48:1–12, 2001.

[dBvKOS97] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[FHS96] J. Friedman, J. Hershberger, and J. Snoeyink. Efficiently planning compliant motion in the plane. *SIAM J. Comput.*, 25:562–599, 1996.

[FM01] S. P. Fekete and J. S. B. Mitchell. Terrain decomposition and layered manufacturing. *International Journal of Computational Geometry and Applications*, 11:647–668, 2001.

[HS95] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.

[HT85] M. Houle and G. Toussaint. Computing the width of a set. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 1–7, 1985.

[IJM+01] I. Ilinkin, R. Janardan, J. Majhi, J. Schwerdt, M. Smid, and R. Sriram. A decomposition-based approach to layered manufacturing. In *Proc. 7th Workshop Algorithms Data Struct.*, volume 2125 of *Lecture Notes Comput. Sci.*, pages 389–400. Springer-Verlag, 2001. To appear in *Computational Geometry: Theory and Applications*.

[Jac92] P. Jacobs. *Rapid Prototyping & Manufacturing: Fundamentals of StereoLithography*. McGraw-Hill, 1992.

[KF98] Chua Chee Kai and Leong Kah Fai. *Rapid Prototyping: Principles and Applications in Manufacturing*. Wiley and Sons, Inc., 1998.

[MJS+99] J. Majhi, R. Janardan, J. Schwerdt, M. Smid, and P. Gupta. Minimizing support structures and trapped area in two-dimensional layered manufacturing. *Computational Geometry: Theory & Applications*, 12:241–267, 1999.

[MJSG99] J. Majhi, R. Janardan, M. Smid, and P. Gupta. On some geometric optimization problems in layered manufacturing. *Computational Geometry: Theory & Applications*, 12:219–239, 1999.

[MJSS98] J. Majhi, R. Janardan, J. Schwerdt, and M. Smid. Multi-criteria optimization algorithms for layered manufacturing. In *Proceedings of the 14th Annual ACM Symposium on Computational Geometry*, pages 19–28, 1998. To appear in *International Journal of Mathematical Algorithms*.

[Ove81] M. Overmars. Dynamization of order decomposable set problems. *J. Algorithms*, 2:245–260, 1981. Corrigendum in 4(1983), 301.

[OvL81] M. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[PS85] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[Tou83] G. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON '83*, pages A10.02/1–4, 1983.