Computing the Straight Skeleton of a Simple Polygon from its Motorcycle Graph in Deterministic $O(n \log n)$ Time.

John C. Bowers*

Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA. jbowers@cs.umass.edu

Abstract. We give the first deterministic $O(n \log n)$ time algorithm for computing the straight-skeleton of a simple polygon given its induced motorcycle graph as input. The previous best takes expected $O(n \log^2 n)$ time. Our algorithm is reminiscent of Shamos and Hoey's divide and conquer algorithm for computing the Voronoi diagram of a set of planar points. Currently, the fastest algorithms for computing the straight-skeleton of a simple polygon (not given its motorcycle graph as input) are: an *expected* $O(n^{4/3+\epsilon})$ time algorithm which first computes the motorcycle graph and then finds the straight-skeleton by post-processing and a *deterministic* $O(n^{17/11+\epsilon})$ time algorithm which computes the straight-skeleton directly. As a consequence of our result, computing the straight skeleton of a simple polygon is improved to *deterministic* $O(n^{4/3+\epsilon})$ time.

1 Introduction

The straight skeleton of a simple polygon (Fig. 1b) is a tree-like structure that subdivides its interior into regions. It was first defined by Aichholzer et al. in [1] by tracing the vertices of the polygon during a wavefront process in which the sides of the polygon are moved inwards in parallel at constant speed. The trace of the vertices during the wavefront process forms the straight-skeleton. It has found a wide array of applications including polygon interpolation [3], procedural modeling of urban environments [14], biomedical imaging [5], and polygon decomposition [13] to name just a few. For convex polygons, the straight skeleton is identical to the medial axis and is linear time computable, but for general simple polygons the computational complexity is still an open problem. The current theoretically fastest algorithms for computing the straight skeleton first compute a structure called the induced motorcycle graph, which was introduced by Eppstein and Erickson [7], and then compute the straight skeleton as a post-processing step. A motorcycle graph is given by placing "motorcycles" in the plane at initial positions with different constant velocities. Each motorcycle moves according to its velocity while laying down a track behind it and crashes if it hits another track. See Fig. 1c.

* Research supported by an NSF graduate fellowship under Grant No. S121000000211.

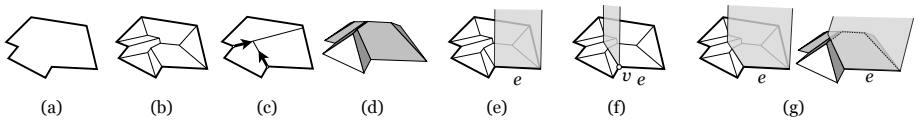

Fig. 1: (a) A polygon. (b) Its straight skeleton. (c) The induced motorcycle graph. (d) The straight skeleton roof. (e) An edge slab. (f) The motorcycle slab for v with respect to e. (g) Shows a view of slab(e), which is the union of the edge and motorcycle slabs for e from $z = +\infty$ (left) and in perspective (right).

Contribution. The purpose of this paper is to present a deterministic $O(n \log n)$ time algorithm proving:

Theorem 1. *There is an algorithm computing the straight-skeleton of a simple polygon P with n vertices from its induced motorcycle graph in $O(n \log n)$ time and $O(n)$ space.*

This is the first *deterministic* $O(n \text{ polylog}(n))$ time algorithm and improves on the previous best due to Cheng and Vigneron [4] which takes *expected* $O(n \log^2 n)$ time. As a side effect, using our algorithm as a post-processing step to the motorcycle graph algorithm of Vigneron and Yan [15] improves the current best algorithms for computing the straight skeleton of a simple polygon from expected $O(n^{4/3+\epsilon})$ [15] and deterministic $O(n^{17/11+\epsilon})$ [7] to deterministic $O(n^{4/3+\epsilon})$ time.

Corollary 1. *There is an algorithm computing the straight skeleton of a simple polygon in deterministic $O(n^{4/3+\epsilon})$ time.*

Prior Work. The first sub-quadratic straight skeleton algorithm is due to Eppstein and Erickson [7] and takes $O(n^{17/11+\epsilon})$ time. This is the fastest for more general planar straight line graphs (PSLGs) and the fastest *deterministic* algorithm for polygons (with or without holes). They introduced *motorcycle graphs* as an abstraction of the main difficulty, but did not give an algorithm for straight skeletons that uses motorcycle graph as input. The first such algorithm was described by Cheng and Vigneron [4]. They give an algorithm computing a motorcycle graph in $O(n^{3/2} \log n)$ time and a post-processing step computing the straight skeleton of a polygon with h holes from its motorcycle graph in expected $O(n\sqrt{h} \log^2 n)$ time. The first step was recently improved to $O(n^{4/3+\epsilon})$ time by Vigneron and Yan [15]. Together with the post-processing step of [4], this leads to *expected* $O(n^{4/3+\epsilon})$ time for computing straight-skeletons of simple polygons. The best known lower bounds for straight skeletons are $\Omega(n \log n)$ for PSLGs [7] and polygons with holes [9], and $\Omega(n)$ for simple polygons. A parallel thread of research focuses on algorithms which perform better in practice than their theoretical upper bounds. Huber and Held [10], describe an $O(n^2 \log n)$ time algorithm for computing the straight skeleton of planar straight-line graphs that uses the motorcycle graph which behaves like $O(n \log n)$ in practice–though worst case examples can be constructed. Similarly, Palfrader et al., [11] investigate the algorithm from [2] and show that it behaves like $O(n \log n)$ in practice, though examples requiring $O(n^2 \log n)$ are known. It remains open to close the gap between theoretical upper and lower bounds and experimental observation.

2 Preliminary Terms

Motorcycle graphs. A motorcycle graph is defined for a set of motorcycles placed at points $p_1, \ldots, p_n$ in the plane each with a velocity vector $v_1, \ldots, v_n$. A motorcycle M_i begins at p_i at time $t = 0$ and moves along the ray $p_i + tv_i$, leaving a track behind it. Motorcycle M_i moves with velocity v_i and crashes if it encounters another motorcycle's track. The *motorcycle graph* is given by vertices for the initial positions $p_1, \ldots, p_n$ and the crash sites $c_1, \ldots, c_n$ for each motorcycle, and an edge for each track. The *motorcycle graph induced by a polygon* P, denoted $MG(P)$, is given by creating a motorcycle for each reflex vertex v of the polygon, with speed equal to $1/\sin(\theta/2)$, where θ is the interior angle at v in P. This speed is the same as the speed a vertex of the wavefront moves in the straight skeleton computation. In addition to the tracks, the polygon edges are treated as obstacles and a motorcycle crashes if it encounters either an edge or a track. See Fig. 1c.

The roof model of the straight skeleton. The straight-skeleton can be defined by tracing the vertices of a wavefront process in which the sides of the polygon are moved inwards in 2D. An alternative view of the straight skeleton is the *roof model* [1]. In the roof model we view the straight skeleton as a polygonal "roof" of faces in $\mathbf{R}^3$ each lying in the upper half space $z \geq 0$ with the boundary edges embedded in the xy-plane. The roof model is given by lifting each vertex v of the straight skeleton by augmenting its position with a z-coordinate equal to the time t at which the wavefront reaches v. We call this the *straight skeleton roof*, denoted $R(P)$. The non-boundary edges of the roof is the (lifted) straight-skeleton, denoted $SS(P)$. See Fig. 1d. Each face of the roof lies in a plane through its base edge making a dihedral angle of $\pi/4$ with the xy-plane.

Edge and motorcycle slabs. An alternative characterization of $R(P)$ is given in [4]. There $R(P)$ is defined as the lower envelope of a set of partially infinite strips in $\mathbf{R}^3$ called slabs defined with respect to the edges of the polygon P and the edges of the motorcycle graph $MG(P)$. For each edge e of P they define an *edge slab* and for each reflex vertex v of P they define two *motorcycle slabs*, one for each edge incident v. Before defining the slabs, let us attach a coordinate frame to each edge of P. Define three unit 3-vectors along e: an *edge vector* $\boldsymbol{E}_e$, a *slope vector* $\boldsymbol{S}_e$, and a *normal vector* $\boldsymbol{N}_e$. Given an edge e of P, $\boldsymbol{E}_e$ is the unit vector pointing along e in counter-clockwise direction around P; $\boldsymbol{S}_e$ is the unit vector orthogonal to $\boldsymbol{E}_e$ lying above the interior of P and making a dihedral angle of $\pi/4$ with the xy-plane; and $\boldsymbol{N}_e = \boldsymbol{E}_e \times \boldsymbol{S}_e$. The *edge slab* of an edge e is defined by $\{p + t\boldsymbol{S}_e \,|\, p \in e, t \geq 0\}$. Let u be a reflex vertex of P and M_u be its motorcycle in $MG(P)$, c_u be the crash site of M_u, and t_u be the crash time. Lift c_u into $\mathbf{R}^3$ to obtain $\bar{c}_u$ by augmenting t_u as its z-coordinate. Let e be an edge of P incident u. Then the *motorcycle slab for u with respect to e* are the points $\{p + t\boldsymbol{S}_e \,|\, p \in (u, \bar{c}_u), t \geq 0\}$ where p is on the line segment $(u, \bar{c}_u)$. We call $(u, \bar{c}_u)$ the *lifted motorcycle track*. See Fig. 1e, f.

The structure slabs(P)***.*** Each edge e has one edge slab and for both of its endpoints it has a motorcycle slab if the endpoint is reflex in the polygon. All slabs for e are contained in the plane through e with normal $\boldsymbol{N}_e$. As in [10] we simplify the notation by referring to the union of the edge slab and any motorcycle slabs

for an edge e as *the slab for* e, denoted $\text{slab}(e)$. See Fig. 1g. We denote the set of slabs for all edges of the polygon by $\text{slabs}(P)$ (i.e. $\text{slabs}(P) = \{\text{slab}(e) \mid e \in P\}$). The lower envelope of $\text{slabs}(P)$ is given by keeping the part of each slab which is lower (in terms of z-coordinate) than all other slabs. In [4] it is shown that (1) $R(P)$ is equivalent to the part of the lower envelope of $\text{slabs}(P)$ which projects orthogonally onto the interior of P and (2) the face with base edge e can be defined as the lower envelope in the direction of $\boldsymbol{S}_e$ in the plane supporting $\text{slab}(e)$ of the line segments given by intersecting all other slabs with $\text{slab}(e)$. We call (2) the *local (2D) definition* for a face of the straight skeleton roof and use these two characterizations in the remainder of the paper.

Assumptions. We assume real-RAM computation and that the input is *non-degenerate*, meaning no two motorcycles crash simultaneously. To simplify the exposition, we also assume that the polygon is in *general position*, meaning no two edges are collinear and no four slabs meet at a point, *but show in the appendix how to remove this assumption while maintaining the same time bounds* which makes our assumptions match those of the randomized $O(n \log^2 n)$ time algorithm from [4]. Note, however, that it is not trivial to remove the non-degeneracy assumption, and the current best algorithm for the general case is the $O(n^{17/11+\epsilon})$ approach of [7].

3 Overview

We prove Theorem 1 by giving a divide and conquer algorithm reminiscent of Shamos and Hoey's $O(n \log n)$ time Voronoi diagram algorithm [12]. Our algorithm operates in the roof model (see Sec. 2) of the straight skeleton. The basic idea is to define a roof model for sub-chains of the polygon P and then perform the following: divide P into two equal length sub-chains, recursively compute roofs for the two sub-chains, and then merge the two sub-chain roofs into a roof for P. The main difficulty is in choosing an appropriate roof model for sub-chains and a merge operation. Our roof model, which we call a *partial roof* is presented in Sec. 4. The merge operation is presented in Sec. 5. The proof of Thm. 1 relies on:

Lemma 1 (Linear complexity of partial roofs). *The combinatorial complexity of a partial roof for an n-length sub-chain of a simple polygon is $O(n)$.*

Lemma 2 (Merging partial roofs in linear time). *Given two partial roofs with base chains that are co-incident sub-chains of a simple polygon P, there is an algorithm for computing a partial roof of the concatenated base-chains in $O(n)$ time.*

Lemma 3 (A partial roof of the entire polygon is the straight skeleton). *Let P be a simple polygon, $R(P)$ be its straight skeleton roof, and R be a partial roof for P. Then $R(P) = R$.*

The remainder of this extended abstract defines partial roofs, the merge operation, and the straight skeleton roof computation and a sketch of the correctness proof. Omitted details are in the appendix.

4 Partial roofs

Recall that the straight skeleton roof $R(P)$ is given by the restriction of the lower envelope of the set of slabs of the polygon, slabs(P), to the interior of the polygon (see Sec. 2). A straightforward divide and conquer algorithm computing $R(P)$ is to subdivide P into equal length chains C_1 and C_2, recursively compute the lower envelope of slabs(C_1) and slabs(C_2), and merge to obtain the lower envelope of slabs(P). There are two problems with this: (1) the combinatorial complexity of the lower envelopes may be $\Omega(n^2\alpha(n))$[1] [6], and (2) merging lower envelopes of slabs is in general non-trivial. However, not all parts of the lower envelopes of slabs(C_1) and slabs(C_2) can possibly appear in $R(P)$. An edge of P is associated with only one face of $R(P)$ but its slab may appear as multiple faces in the lower envelope of slabs(C_1). This motivates our definition of a *partial roof*. The partial roof is closely related to the lower envelope, although its elements are not necessarily contained in elements of the lower envelope. For example, Fig. 2b shows the lower envelope of slabs for a three-edge chain, and Fig. 2c shows a partial roof.

Overview. Like the straight skeleton roof, the *partial roof* is a piecewise linear surface in $\mathbf{R}^3$ made of vertices, edges, and faces. It is defined for a sub-chain of a polygon. Each sub-chain edge is the *base edge* of exactly one face in the partial roof. Unlike the final straight skeleton roof, however, a partial roof may have unbounded faces where part of the face stretches out to infinity. Each face is a simple (possibly unbounded) polygon that lies in the slab for its base edge and is monotone with respect to its base edge. Given a face f with base edge e, any unbounded edge in f is a ray emanating from a vertex of f in the direction of the slope vector of slab(e), $\boldsymbol{S}_e$ (see Sec. 2). In the partial roof in Fig. 2c the face with base edge e_1 is unbounded and the other two are bounded. In addition to the base edges we have one other special type of edge: the edges incident to a base edge at a vertex which is reflex in the polygon are called *motorcycle edges* and are contained in the vertex's lifted motorcycle track (Sec. 2). The face f_1 in Fig. 3 has a motorcycle edge v_1v_2.

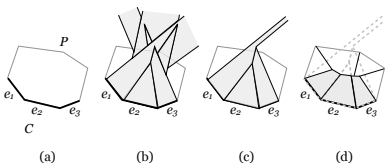

Fig. 2: (a) A polygon P and a sub-chain C of P with edges e_1, e_2, and e_3. (b) The lower envelope of slabs(C) restricted to the $z \geq 0$ half-space (with seven faces) which contains only the part of each slab visible from $z = -\infty$. (c) A partial roof for C (with only three faces). (d) The relationship between faces of a partial roof and faces of the final roof: the partial roof face (dotted outline) for each e_i on the sub-chain contains its corresponding final face (shaded).

Combinatorial description. Combinatorially, a partial roof is a planar graph with a well defined outer face[2] whose boundary is a cycle. The partial roof minus the outer face is topologically a disk. Unbounded faces are "closed" by adding dummy vertices at infinity. We label each vertex with its *position*, which is either ∞ for the dummy vertices, or else is a point in $\mathbf{R}^3$. We use vertical bars $|\cdot|$ to

[1] Where $\alpha(n)$ denotes the inverse Ackermann function.

[2] In the remainder, we do not include the outer face when counting the faces.

denote the realization of any vertex, edge, face, or roof (e.g. the realization of a partial roof R is denoted $|R|$). We denote the boundary edges (those incident the outer face) by ∂R. The boundary is divided into two sub-chains, the *base chain* and the *fringe chain*. The base chain, denoted base(R), consists of all of the base edges. The fringe chain, denoted fringe(R), contains the remaining edges incident to the outer face, and contains all of the infinite dummy vertices. The fringe chain satisfies the following *fringe length invariant*: besides the vertices incident to the first and last edges, the only degree two vertices are infinite dummy vertices and no two are consecutive. From this we have:

Proof of Lemma 1. Since each face has a base chain edge, a face can be incident at most 2 fringe vertices of degree > 2 (else we can find a face not incident to the base chain) and by the fringe length invariant each face is incident to at most 3 fringe vertices of degree 2. Since the internal vertices are of degree 3, and the partial roof is a planar graph with n faces, then the lemma follows by standard counting arguments. □

Main invariants. Each partial roof satisfies the following two invariants. Both are relationships between elements of any partial roof and elements of the final straight skeleton roof. The first, which we call the *face containment invariant*, is that for every edge of the base chain, its corresponding face in the partial roof contains its corresponding face of the straight skeleton roof (in $\mathbf{R}^3$). This property is illustrated in Fig. 2d. The second, which we call the *edge containment invariant*, is that for every pair of edges on the base chain, if the corresponding pair of faces in the final roof are incident along an edge, say e', then the corresponding pair of faces in the partial roof are also incident along an edge, say e, and the realization of e contains the realization of e' (in $\mathbf{R}^3$). We now prove:

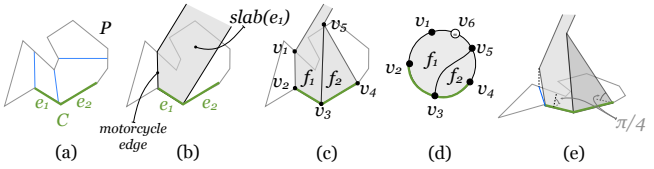

Fig. 3: (a) A polygon P and sub-chain C (green) with two edges e_1 and e_2. The induced motorcycle graph is drawn in blue. (b) A view of slab(e_1) from $z = +\infty$. The slab has a motorcycle edge at the reflex vertex of e_1. (c) A partial roof for C viewed from $z = +\infty$. (d) The combinatorial representation of the partial roof. The infinite part of f_1 is represented by the dummy vertex v_6. (e) A 3D perspective view.

Proof of Lemma 3 (Sec. 3). Let e_1 be an edge of P. Then e_1 is the base edge of a face f_1 in R and a face f_1' in $R(P)$. We first claim that for each edge of f_1' there is a corresponding edge of f_1 which is equal to it (in $\mathbf{R}^3$). Let e' be any edge of f_1' which is not the base edge. Then there is a second face f_2' of $R(P)$ incident to e'. Denote its base edge by e_2 and let f_2 denote the corresponding face in R. By the edge containment property there must exist an edge e in R which is incident to both f_1 and f_2 such that its realization $|e|$ contains the realization $|e'|$. Further, if $|e|$ strictly contains $|e'|$, then f_1 is not simple because the edges incident e' also have corresponding edges in f_1 that contain them and one must be crossed by $|e|$, a contradiction. Thus $|e| = |e'|$. It follows that the faces f and f' are identical. □

Lemma 4 (Existence). *For any polygon base-chain, there exists a partial roof.*

Proof. Start with the straight-skeleton roof $R(P)$ and delete the faces whose base edge is not on the sub-chain C, producing R. The boundary of R consists of C, and a fringe chain of edges which are internal in $R(P)$. For any face, the set of all edges incident the fringe forms a connected chain $e_1, \ldots, e_m$, which may start or end with a motorcycle edge[3]. Ignoring the motorcycle edges (if they exist), replace the chain with two edges connected at an infinite vertex. This removes only edges on the fringe whose base label contains an edge not on C. Each face is extended out to infinity in the direction of monotonicity. Thus R is a partial roof for C. Note that if C is a single edge, then R is exactly the edge's slab. □

5 Merging partial roofs

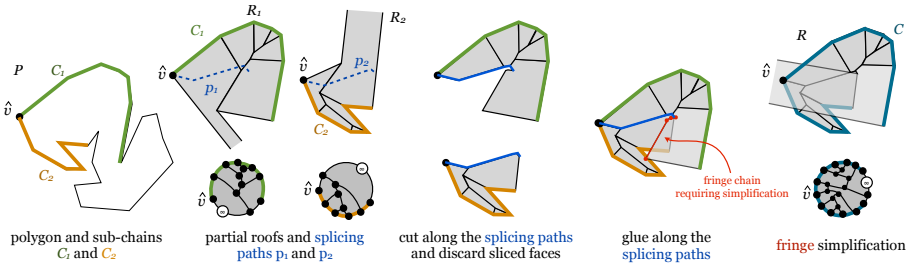

Fig. 4: A merge operation on partial roofs R_1 and R_2 for sub-chains C_1 and C_2 producing R. In reality $|R_1|$ and $|R_2|$ overlap, but are drawn separate for visualization purposes. The splicing paths p_1 and p_2 represent the intersection path of $|R_1|$ with $|R_2|$ beginning at $\hat{v}$. The view is from above with overlapping faces drawn with transparency. The disks underneath the roofs depict the combinatorial representation with white vertices as infinite dummy vertices.

Procedure. The merge operation takes as input two partial roofs for co-incident sub-chains of a simple polygon and produces a partial roof for the the combined sub-chain. It starts at the *gluing vertex* $\hat{v}$ common to both sub-chains. It then performs the following. (1) Walk along the intersection of the two roofs until hitting a boundary edge or previously visited face. We call the path traced by the walk on each partial roof the *splicing path*. (2) Subdivide the faces along the splicing path cutting each into two pieces. For each subdivided face, one piece is "beneath" the other partial roof in the z-direction and the other is "above" it. Discard the piece that lies above the other partial roof. The result of the cutting and discarding is that the splicing path becomes part of the boundary of both partial roofs. (3) Glue the two resulting surfaces together along the splicing path. See Fig. 4. Finally (4) Perform a "clean-up" operation to ensure that the fringe chain maintains the fringe length invariant. Figure 4 illustrates a single merge. For (1) we use a ray-shooting technique, which we discuss in the proof

[3] The connectedness proof is trivial, and for space we leave it to the appendix (see Lem. 9)

of Lem. 8. Step (3) is a common operation on piecewise linear surfaces. We now give more details on steps (2) and (4):

Subdividing the faces. For most faces the splicing path traverses the entire face, and the subdividing the face along the splicing path is well defined. The only special case is the last face encountered. If the splicing path does not simultaneously encounter a boundary edge in both partial roofs, then in one of the roofs, say R_2, the last face f encountered by the path is not completely cut into two. Let x be the endpoint of the splicing path in f. In the local coordinate system of $|f|$, start at $|x|$ and trace the ray emanating from $|x|$ in the direction of the slope vector $\boldsymbol{S}_{\text{base}(f)}$ (Sec. 2). This either hits an edge of $|f|$ or escapes to ∞. In the first case, split the hit edge at the hit point by adding a vertex y and subdivide f by p and an edge from x to y. Otherwise, f must have an infinite vertex, say v_∞. Split f by cutting along p, and then adding an edge from x to v_∞. See Fig. 5.

Fringe simplification. Discarding split faces in Step (2) may result in edges being added to the fringe chain that were previously internal to one of the partial roofs resulting in a violation of the fringe length invariant. In order to maintain the invariant, we perform the following clean-up. The edges of each face on the boundary of the resulting surface R which are not base or motorcycle edges form a connected chain $e_1, \ldots, e_m$ from a vertex u to a vertex v[4]. Each interior vertex of the chain has degree 2. If $m \geq 2$ we replace the chain by adding a new infinite dummy vertex w and swapping out the chain with two edges uw and wv.

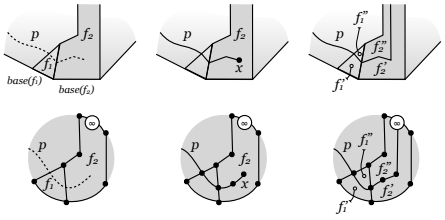

Fig. 5: An example of the subdivision step in two faces f_1 and f_2. The top depicts the geometric realization of a partial roof and the bottom depicts the combinatorial representation. *Left:* The dotted line represents the splicing path p. *Middle:* p ends at a point x in f_2. *Right:* Subdividing along p.

5.1 Correctness and analysis

To analyze the algorithm and prove correctness, we need the following property.

Lemma 5. *The part of the splicing path along each face is monotone w.r.t. the base edge of the face.*

This property is used in Lem. 8 to compute the splicing path in linear time and is used in the proof of Lem. 6, which is needed for correctness. The proof is a modification of the proof, due to Aichholzer et al. that the faces of the roof model of a bisector graph are monotone with respect to their base edges ([1, Lem. 4]). We reproduce their proof in appendix with the modifications necessary for our setting.

Correctness. To prove the correctness of the algorithm, we need to show that the output surface (1) each face is incident along exactly one edge of the base

[4] The proof of this is straightforward, and for space is left to the appendix (Lem. 9)

chain and vice versa, (2) is topologically a disk, and (3) maintains the invariant properties for partial roofs. The first is trivial: the splicing path cannot cut through a base edge, since each slab is only incident the xy-plane at a base edge, the splicing path lies on the intersection of slabs, and the polygon is simple. Also, we always discard the upper part of each face cut by the splicing path, and thus keep the part containing its base edge. For the second, we need the following:

Lemma 6. *The set of faces discarded from each partial roof in step (2) of the merge operation form a topological disk with the splicing path along its boundary.*

The proof uses the monotonicity of the splicing path (Lem. 5) to define a direction for each base edge incident to a face that is traversed by the splicing path. This direction encodes which side of the splicing path the base edge lies. We show by case analysis that on an input partial roof, all such directed base edges point in the same direction along the boundary. From this, and the genericity assumption, which ensures the only vertex of an input partial roof hit by the splicing path is the initial gluing vertex, the lemma follows. For space reasons, we include the proof in the appendix. As a direct corollary:

Corollary 2. *The surface produced by a merge operation is topologically a disk.*

Proof. Since the splicing path starts at the end of the base chain, the first (split) face discarded along the splicing path is incident to the boundary along an edge. By Lem. 6, the discarded faces form a disk. Therefore, after discarding the faces from each partial roof we are left with two topological disks, and the splicing path is a connected sub-chain along the boundary of each. Gluing the two together along this sub-chain thus produces a single disk. □

To complete the correctness of the merge operation we need two main ingredients. First, we need to prove the face containment invariant. This essentially follows by a proof by contradiction using the fact that the operation "cuts down" faces by intersecting them with other faces which are also contained within slabs. The second is the edge containment invariant, which has two parts, first, that certain edges of the partial roof *exist*, and second that they geometrically contain corresponding edges in the final roof. The geometric containment essentially follows the same proof by contradiction as for faces. To prove existence we use an inductive argument on certain paths in the final straight skeleton roof. We now provide more detail:

Lemma 7. *The merge operation correctly computes a partial roof.*

Proof. Let R denote the output of the merge operation on partial roofs R_1 and R_2 and $R(P)$ denote the final straight skeleton roof. We first prove the face containment invariant. Note that thus far we have used the 3D lower-envelope definition of the partial roof, but for proving the invariant it is conceptually convenient to reason in the *local (2D) definition* of each face of $R(P)$ (defined in Sec. 2). As a reminder: given a base edge e, the face f for e is *locally defined* by first intersecting all other slabs with slab(e) to obtain a set of line segments and

then computing the lower envelope of these segments with respect to the vector $\boldsymbol{S}_e$. Now assume that some face f' of R violates the face containment invariant. Then there is an edge of f' lies on the interior of the corresponding face f in the straight skeleton roof. That edge represents an intersection between two slabs, contradicting that f is the lower envelope of such intersections.

We now prove the edge containment invariant. Let e_1 and e_2 be base edges of R such that the corresponding faces f_1' and f_2' in $R(P)$ are incident along an edge e'. There are two cases: either e_1 and e_2 are both edges of C_1 (resp. C_2), or one is an edge of C_1 and the other is an edge of C_2.

Case 1: By the edge containment invariant on R_1, there is an edge e in R_1 incident to the faces of R_1 with base edges e_1 and e_2 such that $|e|$ contains $|e'|$. For contradiction, suppose that no such edge exists in R. Then the face with base edge e_1 in R_1 is cut by the splicing path so that e is part of the discarded face. We then have the same contradiction as for the face containment invariant.

Case 2: *Wlog* let e_1 be an edge of C_1 and e_2 be an edge of C_2. We now prove that the faces f_1 and f_2 in R with base edges e_1 and e_2 are incident along some edge e and $|e|$ contains $|e'|$. Since the edges of the straight skeleton form a tree, there exists a unique path p' along the interior edges of the straight skeleton roof $R(P)$ from $\hat{v}$ to e'. We claim that p' corresponds to the first part of the splicing path p. Let k be the length of p'. The proof is by induction for i from 1 to k.

Base step: By definition, the first edge of both p' and p is along the intersection of the slabs of the base edges incident to $\hat{v}$. Geometric containment follows the argument as above.

Inductive step: Now assume the claim is true for the first $i < k$ edges of p'. Denote the edges of p and p' in order from $\hat{v}$ by $p_1, p_2, \ldots$ and $p_1', p_2', \ldots$, resp. Let $\bar{f}_1'$ and $\bar{f}_2'$ denote the faces incident to p_i' and $\bar{f}_1$ and $\bar{f}_2$ be the faces of R with the same base edges. Then $\bar{f}_1$ and $\bar{f}_2$ are incident along p_i and $|p_i|$ contains $|p_i'|$. We prove that this holds for $i{+}1$. We first need to prove that p contains an edge with index $i+1$. Let v be the vertex between p_i' and p_{i+1}'. By genericity there is one other internal edge, say d', that is also incident to v. Denote the faces incident to p_{i+1}' by $\bar{f}_1'$ and $\bar{f}_2'$ such that the base edges are on C_1 and C_2 (resp.). Without loss of generality assume that d' is incident to $\bar{f}_2'$. Let $\bar{f}_3'$ be the other face incident to d'. See Fig. 6. Since $\bar{f}_2'$ and $\bar{f}_3'$ lie on the same side of p', by Lem. 6 the base edge of $\bar{f}_3'$ is on C_2. By the edge containment invariant on R_2, there is an edge d in R_2 which is incident to two faces with base edges equal to the base edges of $\bar{f}_2'$ and $\bar{f}_3'$. Let $\bar{f}_1$, $\bar{f}_2$, and $\bar{f}_3$ be the faces of R with base edges corresponding to $\bar{f}_1'$, $\bar{f}_2'$, and $\bar{f}_3'$ (resp.). By Case 1 above, d is an edge of R between $\bar{f}_2$ and $\bar{f}_3$. Since d borders $\bar{f}_3$ and $|d|$ contains $|d'|$, then the splicing path between $\bar{f}_1$ and $\bar{f}_3$ must hit d at f_2. Since d is not a boundary edge, the splicing path continues along the intersection of $\bar{f}_1$ and $\bar{f}_2$. This edge

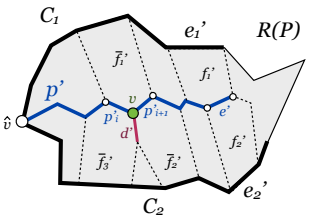

Fig. 6: The setup in the straight skeleton roof for the inductive step of the edge containment invariant.

is p_{i+1}. Geometric containment then follows by the same proof by contradiction as above. □

Existence of a partial roof follows from the observation that for any edge e of a simple polygon P, slab(e) is itself a partial roof for e. That a partial roof exists for any sub-chain C now follows by induction.

Running time. We store each partial roof as a doubly-connected edge list ([8]) which handles most of the operations we need efficiently out-of-the-box. The only non-trivial part is finding the splicing path. We compute the splicing path by iteratively walking along the intersection between faces. At each step we need to determine which face the splicing path exits next along the walk. Together with Cor. 2 and Lem. 7, the following completes the proof of Lem. 2:

Lemma 8. *The splicing path can be computed in linear time.*

Proof. Our approach is to use an iterative ray-shooting scheme across the surface of each face. At each iteration, the walk lies on one face of each. The basic procedure is to shoot a ray across the surface of both intersecting faces to find the first edge of either face hit, then advance the splicing path in both faces along this ray to the closer hit-point. This adds an edge to the splicing path and in one of the partial roofs we cross an edge into a new face. Since we only continue until we hit a boundary edge or a face we have already traversed, the length of the final splicing path is at most n and requires shooting $O(n)$ rays. The difficulty is that in general ray shooting in a polygon is not a constant time operation. To overcome this, we exploit the monotonicity of the splicing path across each face (Lem. 5). We first subdivide each face f of both input partial roofs into trapezoids by extending chords on its interior perpendicular to base(f) from each vertex. Each internal vertex of a partial roof has degree 3 and thus is incident to at most 6 trapezoids. This gives us a bounds of $O(n)$ on the number of trapezoids generated. We now perform the same ray-shooting/walking scheme as above except in the trapezoids. The path now traverses trapezoids, but still cannot cross the same trapezoid twice, and thus its length is still $O(n)$. Shooting a ray in a trapezoid takes $O(1)$ time (just check all edges and take the closest hit), so the total time to compute the splicing path is $O(n)$. □

Proof of Main Theorem. Given the merge operation the procedure for computing the straight-skeleton is surprisingly straightforward: subdivide the polygon into equal length sub-chains, recursively compute a partial roof for each, and merge the results to produce the straight-skeleton roof. Given this procedure, Theorem 1 follows directly from Lems. 1, 2, and 3.

Conclusion. Using our algorithm as a post-processing step, the straight skeleton of a non-degenerate simple polygon can be computed in deterministic $O(n^{4/3+\epsilon})$ time. Straight skeletons are far from solved, however. There still exist large gaps between the current best algorithms and known lower bounds of $\Omega(n)$ for simple polygons, and $\Omega(n \log n)$ for more general PSLGs. Additionally, our method works for non-degenerate simple polygons, and the method of [4] works for non-degenerate simple polygons with holes, but we are not aware of a sub-quadratic

algorithm for straight skeletons of a planar straight line graph which meaningfully uses its induced motorcycle graph. These remain intriguing open problems.

References

1. O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.
2. O. Aichholzer and F. Aurenhammer. Straight skeletons for general polygonal figures. In *Proc. 2^{nd} Ann. Int'l. Computing and Combinatorics Conf. COCOON'96, Lecture Notes in Computer Science*, volume 1090, pages 117–126, Hong Kong, 1996. Springer Verlag. [IIG-Report-Series 423, TU Graz, Austria, 1995].
3. G. Barequet, M. T. Goodrich, A. Levi-Steiner, and D. Steiner. Straight-skeleton based contour interpolation. In *Proc. 14th Symp. on Discrete Algorithms*, SODA '03, pages 119–127, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.
4. S. W. Cheng and A. Vigneron. Motorcycle graphs and straight skeletons. *Algorithmica*, 47(2):159–182, January 2007.
5. F. Cloppet, G. Stamon, and J.-M. Oliva. Angular bisector network, a simplified generalized voronoi diagram: Application to processing complex intersections in biomedical images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):120–128, Jan. 2000.
6. H. Edelsbrunner. The upper envelope of piecewise linear functions: Tight bounds on the number of faces. *Discrete & Computational Geometry*, 4(1):337–343, 1989.
7. D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discr. & Comput. Geom.*, 22(4):569–592, 1999.
8. L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM T. Graphic.*, 4:74–123, 1985.
9. S. Huber. *Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice*. PhD thesis, Universitt Salzburg, Austria, June 2011.
10. S. Huber and M. Held. Theoretical and practical results on straight skeletons of planar straight-line graphs. In *Proc. 27th Symp. on Computational Geometry*, SoCG '11, pages 171–178, New York, NY, USA, 2011. ACM.
11. P. Palfrader, M. Held, and S. Huber. On computing straight skeletons by means of kinetic triangulations. In L. Epstein and P. Ferragina, editors, *Algorithms ESA 2012*, volume 7501 of *Lecture Notes in Computer Science*, pages 766–777. Springer Berlin Heidelberg, 2012.
12. M. I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Symp. on Foundations of Computer Science*, SFCS '75, pages 151–162, Washington, DC, USA, 1975. IEEE Computer Society.
13. M. Tanase and R. C. Veltkamp. Polygon decomposition based on the straight line skeleton. In *Proc. 19th Symp. on Computational Geometry*, SCG '03, pages 58–67, New York, NY, USA, 2003. ACM.
14. C. A. Vanegas, T. Kelly, B. Weber, J. Halatsch, D. G. Aliaga, and P. Müller. Procedural generation of parcels in urban modeling. *Comp. Graph. Forum*, 31(2pt3):681–690, May 2012.
15. A. Vigneron and L. Yan. A faster algorithm for computing motorcycle graphs. In *Proc. 29th Symp. on Computational geometry*, SoCG '13, pages 17–26, New York, NY, USA, 2013. ACM.

A Fringe simplification lemma

Lemma 9. *In step (4) of the merge operation (fringe simplification), the edges of each face on the boundary of R that are not motorcycle or base edges form a connected chain $e_1, \ldots, e_m$.*

Proof. Without loss of generality assume that $e_1, \ldots, e_m$ are given in ccw order along the face so that the base chain is between e_m and e_1 (in ccw order). Assume for contradiction that the list of edges $e_1, \ldots, e_m$ is not connected. Then there is some edge e' between e_1 and e_m (in ccw order) that is incident to another face f'. Now f' must contain a base edge, but any path of boundary edges between the base edge of f and the base edge of f' must contain either e_1 or e_m. Thus the base chain is disconnected, which is a contradiction. □

B Proof of Lemma 5

Assume for contradiction that the statement is false. Then without loss of generality there exists a face f of R_1, which is subdivided by p into f' and f'' s.t. f' contains base(f) and $|f'|$ is not monotone with respect to $|\,\text{base}(f)|$. Since $|f'|$ is not monotone there exist lines in slab(f) which are perpendicular to the line supporting base(f) and which intersect the interior of $|f'|$ then leave the interior at some point x and re-enter again at y. Since $|f|$ is monotone in R_1 (by definition), the points x and y lie on $|p|$. We can choose such a line so that x and y lie on edges of the splicing path which are co-incident at a vertex v. Let e_1 and e_2 be co-incident edges of the splicing path through which such lines pass and let v be the vertex between e_1 and e_2. By definition of the splicing path, e_1 and e_2 correspond to faces f_1 and f_2 in R_2 which are co-incident in R_2 along an edge passing through v. Choose a line L leaving $|f'|$ at a point x along e_1 and re-entering $|f'|$ at a point y along e_2, such that the orthogonal projection of the segment (x, y) of L onto the xy-plane is contained within the orthogonal projection of $|f_1| \cup |f_2|$ onto the xy-plane. Recall that slab(f) makes an angle $\pi/4$ with the xy-plane and L follows the direction of steepest ascent in slab(f). Thus L has slope 1. Now project (x, y) onto $|f_1| \cup |f_2|$. The projection results in a polygonal path p' with one edge along $|f_1|$ and one edge along $|f_2|$ beginning at x and ending at y. Thus the average slope of p' must be 1. However, since all the slabs make the same $\pi/4$ angle with the xy-plane, it is not possible that the intersection of any two slabs have slope greater than or equal to one, a contradiction. □

C Proof of Lemma 6

Recall that by Lemma 5, we know that the splicing path is monotone with respect to the base edge it traverses. We use this to assign a direction to each base edge. First, we direct each edge of the splicing path by the walk along the path starting at the gluing vertex. If we take the vector pointing in this direction

along the splicing path edge, and project it onto the vector through the base edge (where each base edge is oriented counter-clockwise), this gives us a well defined direction, denoted dir(e) along the edge. See Fig. 7. We first prove that this direction, which we call the *splicing path direction*, is consistent along the boundary of each input partial roof (Lem. 10). In other words, for a given input partial roof (either R_1 or R_2), and two different base edges e_1 and e_2 whose incident faces are traversed by the splicing path, the splicing path direction is either clockwise on each or counter-clockwise on each. As a corollary (Cor. 3), it next follows that the set of all faces from each input surface lie on the same side of the splicing path. The proof of Lem. 6 then follows.

Lemma 10 (The splicing direction is consistent along the boundary). *Let e_1 and e_2 be base edges of faces intersected by p_1 in R_1 (resp. p_2 in R_2). Then* dir(e_1) *and* dir(e_2) *both point ccw or both cw on ∂R_1 (resp. ∂R_2).*

Proof. Let f_1 and f_2 be faces of R_1 (resp. R_2) which are visited consecutively by the splicing path p and let e_1 and e_2 be the respective base edges. We claim that dir(e_1) and dir(e_2) point in the same direction. This is extended to non-consecutive cases by simple induction along p.

By Lem. 5, we know that the splicing path is monotone in each face with respect to the base edge so examining any part of p in a face f allows us to determine the splicing direction on base(f).

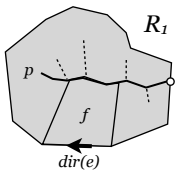

Fig. 7: The splicing path p on $|R_1|$. The gluing vertex is denoted by the white circle. dir(e) for one base edge e of a face f is depicted.

Since f_1 and f_2 are consecutive along p, there is a vertex of p which lies on the edge e between f_1 and f_2. Let v be this vertex and let uv and vw be the edges of p incident v s.t. uv lies on f_1 and vw lies on f_2. By the definition of p there must be some face f_3 of R_2 such that uv lies on the intersection of $|f_3|$ and $|f_1|$ and vw lies on the intersection of $|f_3|$ and $|f_2|$.

We now construct a certain tetrahedron and use it to prove our claim. Let L_1, L_2, and L_3 be the supporting lines of the base edges of $|f_1|$, $|f_2|$, and $|f_3|$. Assuming general position, each pair of lines intersect at a point. Now, let a be the intersection of L_1 and L_2, b be that of L_2 and L_3, and c be that of L_1 and L_3. (Note that this proof extends naturally to the case where L_1 (resp. L_2) is parallel to L_3: instead of letting c be the intersection of L_1 and L_3, we will create a c' and a c'' which are the endpoints of e_1 and e_3 which are farther from L_2. Then wherever we use c in an angle below, simply use the appropriate c' or c'' depending on whether we are talking of a face incident to e_1 or e_3.)

Let T be the tetrahedron with sides abc, abv, bcv, and acv. See Fig. 8a. For each edge of abc project the corresponding slope vector (Sec. 2) onto the xy-plane. Call this the *ascent vector* for each edge of abc. See Fig. 8b.

Each ascent vector points either *inwards* towards the interior of abc or *outwards* towards the exterior of abc. It serves two purposes: the first is that given a base edge of $|f_1|$, $|f_2|$, or $|f_3|$, the ascent vector of its corresponding edge in T

points towards the interior of the polygon P. Call this the *orientation property*. This in turn allows us to determine the counterclockwise direction of travel along a base edge, since it is the direction to the right of the ascent vector; the second is that it allows us to classify the possible types of tetrahedra T.

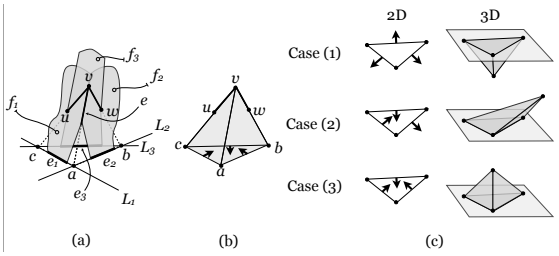

Fig. 8: Illustration of the tetrahedron determined by the three faces in the proof of Lem. 10. (a) Two faces f_1 and f_2 of R_1 and a face f_3 of R_2 meeting along edges uv and vw of the splicing path between R_1 and R_2. The base edges of f_1, f_2, and f_3 are e_1, e_2, and e_3 (resp.) and the supporting lines of the base edges are given by L_1, L_2, and L_3. These lines intersect at a, b and c forming a triangle in the xy-plane. (b) The tetrahedron incident abc with faces lying in the planes supporting f_1, f_2 and f_3. The ascent vectors for each edge of abc are illustrated by bold arrows incident ab, bc, and ac indicating which direction in the xy-plane points "up hill" along the incident face. (c) An example for each of the three possible cases. In 2D we show the triangle abc and its ascent vectors and in 3D we show a tetrahedron realizing those ascent vectors. In Case (1) all ascent vectors point outwards. In Case (2) two ascent vectors point inwards and one outwards. The reverse is also possible (though not depicted). In Case (3) all ascent vectors point inwards.

We classify all possibilities into three classes: (1) all three ascent vectors point outwards, (2) one points inwards and the other two point outwards or vise versa, and (3) all point inwards. See Fig. 8c. We now analyze each case in turn.

Case (1). In this case, it can be shown by elementary geometry that the vertex v lies below the xy-plane, which contradicts the definition p as lying on the intersection of partial roofs (which are made up of parts of edge slabs in the $z \geq 0$ half-space).

Case (2). In this case, it can be shown by elementary geometry that two of the faces of abv, bcv, and acv have an obtuse angle at the vertex which is not common to both faces (e.g. abv and acv are such that b and c are obtuse). There are three such configurations, and we analyze each.

First, assume that $\angle abv$ and $\angle acv$ are obtuse. In this case, the ascent vectors of ab and ac either both point inwards or both point outwards. Let $\boldsymbol{uv}$ and $\boldsymbol{vw}$ denote unit vectors pointing from u to v and v to w (resp.). Project $\boldsymbol{uv}$ onto the edge ac and $\boldsymbol{vu}$ onto the edge ab to obtain $\boldsymbol{uv}^{\perp}$ and $\boldsymbol{vw}^{\perp}$. $\boldsymbol{uv}^{\perp}$ points in the direction from a towards c. $\boldsymbol{vw}^{\perp}$ points in the direction from b towards a. So the two vectors either point both clockwise or both counterclockwise around abc. These two vectors are equivalent to the splicing direction through the base

edges of f_1 and f_2 and by the orientation property of the ascent vectors, both of these will either have a consistent orientation with ab and bc (if both ascent vectors point inwards) or will have the opposite orientation. Either way, the splicing direction along the two base edges is consistent. See Fig. 9a. **Second**, assume that $\angle bav$ and $\angle bcv$ are obtuse. Then we can choose a line L arbitrary close to v in the plane supporting bcv which is perpendicular to L_3 and passes through cv and bv. Since we can choose L arbitrarily close to v, we can ensure that it intersects both uv and vw. This shows, however, that the splicing path p is not monotone in f_3 contradicting Lem. 5. See Fig. 9b. **Third**, assume that $\angle cav$ and $\angle cbv$ are obtuse. This case is symmetric to the previous and results in the same contradiction.

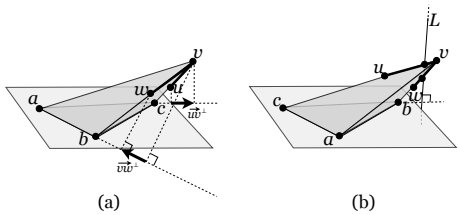

Fig. 9: The setup for the proofs of two sub-cases of Case (2) in Lem. 10. (a) The first sub-case in which the two obtuse angles are $\angle abv$ and $\angle acv$. In this case the direction vectors $\boldsymbol{vw}^{\perp}$ and $\boldsymbol{uv}^{\perp}$ both point clockwise around abc. (b) The case where $\angle cbv$ is obtuse. In this case we can choose a line L in the plane supporting triangle bcv which is perpendicular to the line through bc and crosses both uv and vw.

Case (3). This case follows exactly the same argument as the first argument in Case (2) with the simplification that the ascent vectors always point inwards. □

Base edges lie on the same side of p. The splicing direction $\mathrm{dir}(e)$ encodes the side of the splicing path on which the base edge lies. If the base edge e is to the right of the splicing path, then $\mathrm{dir}(e)$ points ccw, otherwise if to the left, it points cw. Since all direction vectors point the same way (Lem. 10) we have the following:

Corollary 3. *All faces discarded in step (2) from R_1 (resp. R_2) lie to the same side of the splicing path.*

Finally, by the general position assumption, except at the gluing vertex, the splicing path intersects only the interiors of faces and edges of the input partial roofs (if it intersected a vertex, then more than three slabs would be co-incident to the intersection point). Thus, the discarded faces form a chain along one side of the splicing path such that each consecutive pair of faces along the path are incident along an edge. This completes the proof of Lem. 6, assuming P is in general position. We now show how to remove the assumption that P is in general position, while maintaining the same running-time for the merge operation. □

D Relaxing our assumptions

In [4] two assumptions are made for the $O(n \log^2 n)$ expected time algorithm: an explicit assumption that the polygon is non-degenerate, meaning that no two motorcycles crash into one another simultaneously, and an implicit assumption that the roof model of the straight skeleton is well-defined, meaning in particular that the straight-skeleton face incident to a particular base edge can be defined locally by the (2D) lower envelope in the edge's slab of the line segments formed by intersecting all other slabs with it. This second assumption is used to compute a single face of the straight-skeleton independently of the other faces by intersecting all other slabs with the face's slab and then using a segment tree to find the lower envelope of the segments in $O(n \log n)$. This implicitly assumes, however, that the intersection of any two slabs is either empty or a line segment. This assumption may be violated, without also violating the non-degeneracy assumption on the motorcycle graph. See for example Fig. 10.

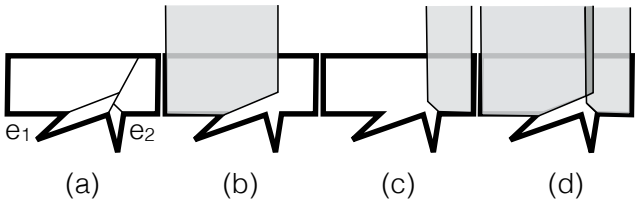

Fig. 10: An example where the local definition of the roof model as a lower envelope of line segments is not well defined. In (a) we have a polygon with its induced motorcycle graph and two collinear edges e_1 and e_2. (b) and (c) show the respective slabs. (d) shows both slabs with the intersection of the two slabs shaded darker. Note that this case occurs even though the motorcycle graph itself is non-degenerate.

In addition to the assumptions made by [4], we have so far assumed that the polygon is in general position, by which we mean that no edges are parallel and no four slabs intersect at a point. We now work to relax these assumptions. We will first show how to remove the general position assumption while maintaining the running time of the algorithm. We will then show how to relax the requirement that no two parallel slabs overlap in a region.

D.1 Removing the general position assumption

We now remove the general position assumption: we allow for parallel slabs (and even co-planar slabs with one restriction) and more than three slabs to intersect at a single point. For the moment, however, let us assume that no two slabs intersect in a degenerate manner: if they intersect at all, the intersection is a line segment or point. In other words, if two slabs are coplanar, then they do not intersect. We will see in the next section how even this assumption can be relaxed. There are now three main difficulties: maintaining the combinatorial complexity

of a partial roof now that more than three slabs may intersect at the same point, maintaining the time bound of the merge operation, now that finding the next face traversed by the splicing path after ray-shooting may require checking all the faces incident to a vertex and more than 6 ray-shooting trapezoids may be incident to a vertex of the partial roof, and maintaining the correctness of the algorithm, since Lem. 10 assumes general position. We now show how to handle each of these difficulties.

Maintaining combinatorial complexity. We required that each internal vertex of a partial roof is of degree 3, which was valid due to the assumption that no four slabs intersected at a common point. This was primarily used in bounding the complexity of a partial roof. However, since each partial roof is a planar graph with n faces and $O(n)$ edges on its boundary, we can relax this restriction to internal vertices of degree *at least* 3. By standard counting arguments on planar graphs, this still maintains the linear complexity of each partial roof with respect to the number of edges on its base chain.

Cutting through a vertex while maintaining the time-complexity of the algorithm. The splicing path may now cut through a vertex v rather than an edge of the input roofs. At any such vertex v, the path will traverse two faces incident to v and all other faces touching v will lie to either side of it. When we cut along the splicing path, the vertex v is duplicated on either side of the cut, and the faces that touch v but are not intersected by the splicing path may be on either side. One possible worry is that when we discard the parts of the faces that are subdivided by the splicing path we may introduce a degeneracy. Figure 11 illustrates this possibility. In the figure f_1 and f_2 are the two faces incident to the splicing path, and both are incident to the fringe. After discarding the parts of f_1 and f_2 on the left side of the splicing path, the face f_3 becomes degenerate and when we glue to the other partial roof the resulting structure is not a disk. However, this situation occurs because f_2 is incident to the fringe, which implies that there is a fringe edge between base(f_2) and base(f_3) on the base chain, a contradiction. Thus, if there is such a face f_3 on the opposite side of the splicing path it is impossible that the face f_2 be incident to the fringe. Similarly no faces whose base edges lie between base(f_2) and base(f_3) can be incident to the fringe, and so the remaining faces intersected by the splicing path cannot introduce a degenerate situation.

The difficulty, then, becomes the analysis of the running time of the algorithm. Under the general position assumption, the splicing path always hit an edge, which allowed us to find the next face traversed by the path in $O(1)$ time. Now, however, if the splicing path passes through a vertex v, we need to test all faces incident to v to find the next face traversed. In the worst case the splicing path will only pass through vertices and never hit edges. A naive worst-case analysis would conclude the splicing path will then require quadratic time. However, with this modification the number of faces checked in total along the splicing path is equal to the sum of the degrees over all the vertices intersected by the splicing path. By a trivial application of the handshaking lemma, this sum is twice the number of edges in the partial roof, which is $O(n)$. We can similarly

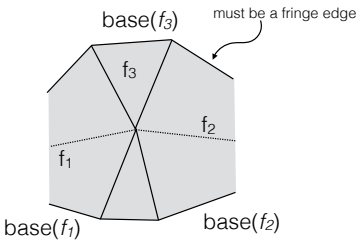

Fig. 11: An example of a situation that cannot occur: the face f_3 lies on the opposite side of the splicing path (dotted) from the base edges of the faces f_1 and f_2 traversed by the splicing path through a vertex, but f_2 is incident to the fringe. This is a contradiction, since it implies that the base chain is disconnected.

bound the number of trapezoids created by the ray-shooting sub-routine: the number of trapezoids incident to any vertex is at most twice the degree of that vertex, so the total number of trapezoids is also $O(n)$. Thus the running time of the merge operation remains $O(n)$ without the general position assumption.

Removing the reliance on Lem. 10. We used Lem. 10 to prove Lem. 6, and Lem. 10 assumes that at a vertex of the splicing path, only three slabs intersect. We could work to extend Lem. 10 by another, but now larger, case analysis involving pyramids rather than tetrahedra. However, we can use the following trick based on a well known, simple observation to avoid the use of Lem. 10 altogether. The observation is that the only valley edges of the straight-skeleton roof–those in which the incident faces slope upwards rather than slope downwards away from the edge, are motorcycle edges of the corresponding slabs. If we look at the part of any splicing path that traces out edges that are part of the final straight-skeleton roof (i.e. those edges along the splicing path used in the induction argument in the proof of correctness), except for the first edge of the path (if it is a motorcycle edge), all the faces from (wlog) R_1 will lie to the right of the path and slope downwards to the base edge, and from R_2 will lie to the left of the path and slope downwards. We can thus use the following check. When we first begin to compute the splicing path, note to which side of it the faces of R_1 and R_2 are sloping downwards (these will necessarily be on opposite sides, and as discussed we ignore the first edge if it is a motorcycle edge). Say that the faces of R_1 lie to the right initially, and the faces of R_2 lie to the left. If at any point we are set to cross into a face such that this property switches: for example the next edge of the splicing path would be across a face of R_1 where the downward slope of the face is towards the left, we simply stop the splicing path there (as we did if we attempted to cross into a faces we had already traversed). This check avoids the need for 10 completely, but maintains the correctness proof of the algorithm, since the potential next edge we have stopped at cannot possibly have the same base label as any edge in the final straight-skeleton roof, by the observation above. The result of Lem. 10 is that this check is superfluous if the input is in general position, since we will never

get to such an edge, but without Lem. 10, this check allows for a more trivial proof of Lem. 6 that works even if the input is not in general position.

D.2 Relaxing the constraints on parallel slabs

In the previous section we showed how to remove the general position assumption on the input, but made the assumption that no two slabs intersect in a degenerate way. We show now that this assumption is not required. We replace it with the following weaker assumption: no pair of coplanar faces in the final straight-skeleton roof are incident to each other. In fact, we have been unable to come up with an example input polygon where this occurs without also having a degenerate motorcycle graph, but so far have not found a proof that having a non-degenerate motorcycle graph implies this case does not occur. We leave this as an interesting open problem.

In any case, the following modification allows our algorithm to work even if slabs intersect in a degenerate way, as long as there are no such degenerate faces in the final straight-skeleton roof. Recall that when we compute the splicing path we stop if we were about to enter a face that has previously been traversed. We now add another similar check, which is essentially the same trick as we used in the previous section to remove the reliance of Lem. 6 on Lem. 10. If the splicing path is about to traverse two faces whose base edges share the same supporting line we stop computing the splicing path at that point. This ensures that we never try to glue together two faces that overlap. Recall that the proof of correctness is an induction starting at the beginning of the splicing path that shows that each edge that "should be" in the partial roof is in the partial roof–it essentially shows that the splicing path walks along a part of the final partial roof first, before going along edges that will eventually be removed. If we stop the splicing path computation due to the check described above, then we have arrived at a point that by assumption cannot possibly be part of the final straight skeleton, and so this shortened splicing path does not contradict the inductive argument. This maintains the correctness of the algorithm while not requiring us to resolve any ambiguities about how to deal with overlapping slabs.