Outer Common Tangents and Nesting of Convex Hulls in Linear Time and Constant Workspace

Mikkel Abrahamsen[*1] **and Bartosz Walczak**[2]

1 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
miab@di.ku.dk

2 **Theoretical Computer Science Department, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland**
walczak@tcs.uj.edu.pl

Abstract

We describe the first algorithm to compute the outer common tangents of two disjoint simple polygons using linear time and only constant workspace. A tangent of a polygon is a line touching the polygon such that all of the polygon lies on the same side of the line. An outer common tangent of two polygons is a tangent of both polygons such that the polygons lie on the same side of the tangent. Each polygon is given as a read-only array of its corners in cyclic order. The algorithm detects if an outer common tangent does not exist, which is the case if and only if the convex hull of one of the polygons is contained in the convex hull of the other. Otherwise, two corners defining an outer common tangent are returned.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases simple polygon, common tangent, optimal algorithm, constant workspace

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.4

1 Introduction

A tangent of a polygon is a line touching the polygon such that all of the polygon lies on the same side of the line. An outer common tangent of two polygons is a tangent of both polygons such that the polygons lie on the same side of the tangent. Two disjoint polygons have exactly two outer common tangents unless their convex hulls are nested. If they are properly nested, there is no outer common tangent. In this paper, we study the problem of computing the outer common tangents of two disjoint simple polygons, each given as a read-only array of its corners in cyclic order. We give an algorithm computing the outer common tangents in linear time using only a constant number of variables each storing a boolean value or an index of a corner in the array. We are therefore working in the *constant workspace model* of computation.

The constant workspace model is a restricted version of the RAM model in which the input is read-only, the output is write-only, and only $O(\log n)$ additional bits of *workspace* (with both read and write access) are available, where n is the size of the input. Clearly, $\Omega(\log n)$ bits in the workspace are necessary to solve any interesting computational problem, because that many bits are required to store an index of or a pointer to an entry in the input. Since blocks of $\Theta(\log n)$ bits are considered to form *words* in the memory, algorithms in the constant workspace model use $O(1)$ words of memory, which explains the name of the model.

* Research partially supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme.

The practical relevance of studying problems in the constant workspace model is increasing, as there are many current and emerging memory technologies where writing can be much more expensive than reading in terms of time and energy [8].

The constant workspace model was first studied explicitly for geometric problems by Asano et al. [4]. Recently, there has been growing interest in algorithms for geometric problems using constant or restricted workspace, see for instance [1, 3, 5, 6, 9, 11, 14].

The problem of computing common tangents of two polygons has received most attention in the case that the polygons are convex. For instance, computing the outer common tangents of disjoint convex polygons is used as a subroutine in the classical divide-and-conquer algorithm for the convex hull of a set of n points in the plane due to Preparata and Hong [17]. They give a naive linear-time algorithm for outer common tangents, as it suffices for an $O(n \log n)$-time convex hull algorithm. The problem is also considered in various dynamic convex hull algorithms [7, 12, 16]. Overmars and van Leeuwen [16] give an $O(\log n)$-time algorithm for computing an outer common tangent of two disjoint convex polygons when a separating line is known, where each polygon has at most n corners. Kirkpatrick and Snoeyink [13] give an $O(\log n)$-time algorithm for the same problem but without using a separating line. Guibas et al. [10] give a lower bound of $\Omega(\log^2 n)$ on the time required to compute an outer common tangent of two intersecting convex polygons, even if they are known to intersect in at most two points. They also describe an algorithm achieving that bound. Toussaint [18] considers the problem of computing separating common tangents of convex polygons and notes that the problem occurs in problems related to visibility, collision avoidance, range fitting, etc. He gives a linear-time algorithm. Guibas et al. [10] give an $O(\log n)$-time algorithm for the same problem. All the above-mentioned algorithms with sublinear running times make essential use of the convexity of the polygons. If the polygons are not convex, a linear-time algorithm can be used to compute the convex hulls before computing the tangents [15]. However, if the polygons are given in read-only memory, $\Omega(n)$ extra bits are required to store the convex hulls, so this approach does not work in the constant workspace model.

Abrahamsen [2] gives a linear-time constant-workspace algorithm to compute the outer common tangents of two simple polygons the convex hulls of which are disjoint. In this paper, we show that the same is possible as long as the polygons (but not necessarily their convex hulls) are disjoint. The algorithm is only slightly different from the one in [2], but its proof of correctness requires much more effort. In particular, the proof relies on an intricate continuous analysis of the algorithm. Before, it was not even clear whether to expect existence of a linear-time constant-workspace algorithm that does not require the convex hulls to be disjoint, because it happens quite often that a computational problem exhibits different behavior for disjoint polygons and for polygons that are not disjoint. For instance, as it has been mentioned above, the outer common tangents of two disjoint convex polygons can be computed in time $O(\log n)$, while doing the same for two convex polygons that intersect in two points requires time $\Omega(\log^2 n)$.

A separating common tangent of two polygons is a tangent of both polygons such that the polygons lie on the opposite sides of the tangent. Two disjoint polygons have exactly two separating common tangents provided that their convex hulls are disjoint. If they intersect properly, there is no separating common tangent. Abrahamsen [2] describes a linear-time constant-workspace algorithm that computes the separating common tangents of two simple polygons. In particular, it detects whether the convex hulls of two simple polygons are disjoint. Our current algorithm can decide whether the convex hulls two simple polygons are nested, which happens when it is unable to find an outer common tangent. To the best of our knowledge, this was not known to be possible in linear time and constant workspace prior to

this work. Our algorithm and the algorithm from [2] together enable us to determine, for two disjoint simple polygons in general position, the full relation between their convex hulls (whether they are nested, overlapping, or disjoint) in linear time and constant workspace.

It remains open whether an outer common tangent of two polygons that are not disjoint can be found in linear time using constant workspace.

2 Terminology and Notation

For any two points a and b in the plane, the closed line segment with endpoints a and b is denoted by ab. When $a \neq b$, the straight line containing a and b that is infinite in both directions is denoted by $\mathcal{L}(a,b)$, and the ray starting at a and going through b is denoted by $\mathcal{R}(a,b)$. For three points a, b, and c, consider the line $\mathcal{L}(a,b)$ as oriented from a towards b, and define $\mathcal{T}(a,b,c)$ to be 1 if c lies to the left of $\mathcal{L}(a,b)$, 0 if a, b, c are collinear, and -1 if c lies to the right of $\mathcal{L}(a,b)$. Let $\mathrm{LHP}(a,b)$ denote the closed half-plane lying to the left of $\mathcal{L}(a,b)$ and $\mathrm{RHP}(a,b)$ denote the closed half-plane lying to the right of $\mathcal{L}(a,b)$.

A *simple polygon*, or just a *polygon*, with *corners* $x_0,\ldots,x_{n-1}$ is a closed polygonal curve in the plane composed of n *edges* $x_0x_1,\ldots,x_{n-2}x_{n-1},x_{n-1}x_0$ such that the segments have no common points other than the common endpoints of pairs of consecutive edges. The region of the plane bounded by a polygon P (including P itself) is a *polygonal region*.

Assume for the rest of this paper that P_0 and P_1 are two disjoint simple polygons with n_0 and n_1 corners, respectively. (We allow one of P_0 and P_1 to be contained in the "interior region" of the other – in that case our algorithm will report that the convex hulls are nested and no outer common tangent exists.) Assume that P_k is defined by a read-only array of its corners $p_k[0],p_k[1],\ldots,p_k[n_k-1]$ for $k\in\{0,1\}$. Assume further, without loss of generality, that the corners of P_0 are given in counterclockwise order and the corners of P_1 are given in clockwise order. (The orientation of a polygon can be easily tested in linear time using constant workspace, and the algorithm can choose to traverse the polygon forwards or backwards, accordingly.) Finally, assume that the corners are in general position in the sense that P_0 and P_1 have no corners in common and the combined set of corners $\{p_0[0],\ldots,p_0[n_0-1],p_1[0],\ldots,p_1[n_1-1]\}$ contains no triple of collinear points.

Indices of the corners of P_k are considered modulo n_k, so that $p_k[i]$ and $p_k[j]$ denote the same corner when $i\equiv j \pmod{n_k}$. For $a,b\in P_k$, the *chain* $P_k[a,b]$ is the portion of P_k from a to b in the order assigned to P_k (counterclockwise for P_0, clockwise for P_1). If i and j are indices of corners on P_k, we write $P_k[i,j]$ to denote $P_k[p_k[i],p_k[j]]$.

A *tangent* of P_k is a line ℓ such that ℓ and P_k are not disjoint and P_k is contained in one of the closed half-planes determined by ℓ. The line ℓ is a *common tangent* of P_0 and P_1 if it is a tangent of both P_0 and P_1. A common tangent is an *outer common tangent* if P_0 and P_1 are on the same side of the tangent, otherwise the common tangent is *separating*.

For a simple polygon P, let $\mathcal{H}(P)$ denote the convex hull of P. The following lemma asserts well-known properties of common tangents of polygons. See Figures 1–3.

▶ **Lemma 1.** *A line is a tangent of a polygon P if and only if it is a tangent of $\mathcal{H}(P)$. Under our general position assumptions, the following holds. If one of $\mathcal{H}(P_0)$ and $\mathcal{H}(P_1)$ is completely contained in the other, there are no outer common tangents of P_0 and P_1. Otherwise, there are two or more, and there are exactly two if P_0 and P_1 are disjoint. If $\mathcal{H}(P_0)$ and $\mathcal{H}(P_1)$ are not disjoint, there are no separating common tangents of P_0 and P_1. Otherwise, there are exactly two.*

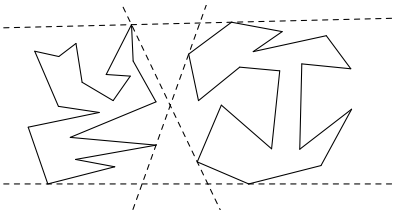

Figure 1 The convex hulls are disjoint – separating and outer common tangents exist.

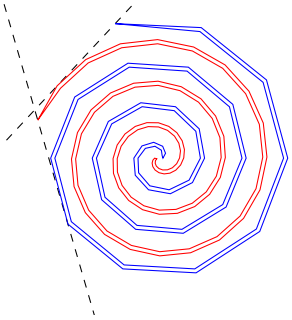

Figure 2 The convex hulls overlap – only outer common tangents exist.

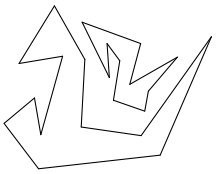

Figure 3 The convex hulls are nested – no common tangents exist.

Algorithm 1: `OuterCommonTangent`(P_0, P_1)

```
1  s_0 ← 0;  v_0 ← 0;  b_0 ← false;  s_1 ← 0;  v_1 ← 0;  b_1 ← false;  u ← 0
2  while s_0 < 2n_0 and s_1 < 2n_1 and (v_0 < s_0 + n_0 or v_1 < s_1 + n_1)
3      v_u ← v_u + 1
4      if T(p_0[s_0], p_1[s_1], p_u[v_u]) = 1
5          if p_{1-u}[s_{1-u}] ∈ Δ(p_u[s_u], p_u[v_u − 1], p_u[v_u])
6              b_u ← true
7          if not b_u
8              s_u ← v_u;  v_{1-u} ← s_{1-u};  b_{1-u} ← false
9      u ← 1 − u
10 if s_0 ≥ 2n_0 or s_1 ≥ 2n_1 or b_0 or b_1
11     return nested
12 return (s_0, s_1)
```

3 Algorithm

Let the outer common tangents of P_0 and P_1 be defined by pairs of corners (ℓ_0, ℓ_1) and (r_0, r_1) so that $\ell_0, r_0 \in P_0$, $\ell_1, r_1 \in P_1$, and $P_0, P_1 \subset \mathrm{LHP}(\ell_0, \ell_1) \cap \mathrm{RHP}(r_0, r_1)$. Algorithm 1 returns a pair of indices (s_0, s_1) such that $(r_0, r_1) = (p_0[s_0], p_1[s_1])$ or, if the convex hulls of P_0 and P_1 are nested so that the tangents do not exist, the algorithm reports that by returning `nested`. Finding (ℓ_0, ℓ_1) requires running Algorithm 1 with the roles of P_0 and P_1 interchanged and with the orders of the corners of P_0 and P_1 reversed – each array reference $p_k[i]$ is translated to $p_{1-k}[-i]$ for $k \in \{0, 1\}$, and the returned result is (s_1, s_0) such that $(\ell_0, \ell_1) = (p_0[s_0], p_1[s_1])$.

The algorithm maintains a pair of indices (s_0, s_1) which determines the *tangent candidate* $\mathcal{L}(p_0[s_0], p_1[s_1])$. Starting from $(s_0, s_1) = (0, 0)$ and advancing the indices s_0, s_1 appropriately, the algorithm attempts to reach a situation that $(p_0[s_0], p_1[s_1]) = (r_0, r_1)$, that is, $P_0, P_1 \subset \mathrm{RHP}(p_0[s_0], p_1[s_1])$. At the start and after each update to (s_0, s_1), the algorithm traverses P_0 and P_1 in parallel with indices (v_0, v_1), starting from $(v_0, v_1) = (s_0, s_1)$ and advancing v_0 and v_1 alternately. The variable $u \in \{0, 1\}$ determines the polygon P_u in which we advance the traversal in a given iteration. If the test in line 4 happens to be positive, then the corner $p_u[v_u]$ lies on the "wrong side" of the tangent candidate, witnessing $P_u \not\subset \mathrm{RHP}(p_0[s_0], p_1[s_1])$.

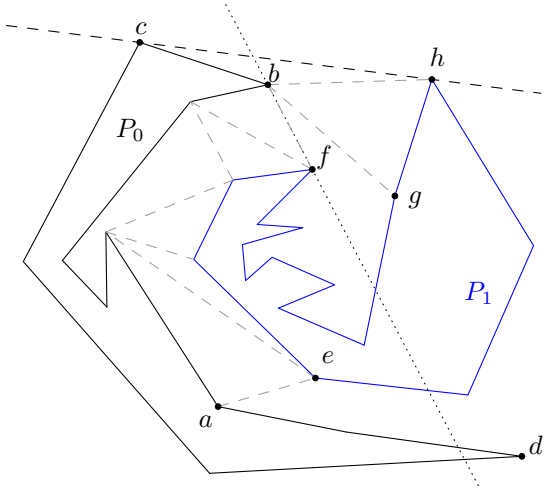

Figure 4 An example of how Algorithm 1 finds the outer common tangent $\mathcal{L}(c, h)$ of P_0 and P_1. The start points are $(p_0[0], p_1[0]) = (a, e)$. The gray dashed line segments are the segments $p_0[s_0]p_1[s_1]$ on the various tangent candidates. In the 11th iteration, an update makes $(p_0[s_0], p_1[s_1]) = (b, f)$, so the tangent candidate becomes the dotted line $\mathcal{L}(b, f)$. In the 19th iteration, $u = 0$ and $p_0[v_0] = d$, so b_0 is set to `true`. In the 28th iteration, $u = 1$ and $p_1[v_1] = g$, and therefore b_0 is cleared. In the 31st iteration, an update makes $(p_0[s_0], p_1[s_1]) = (c, h)$ and the outer common tangent has been found.

In that case, the algorithm updates the tangent candidate by setting $s_u \leftarrow v_u$ and reverts v_{1-u} back to s_{1-u} in line 8, unless a special boolean variable b_u is set, which we will comment on shortly. The reason for reverting v_{1-u} back to s_{1-u} in line 8 is that a corner of P_{1-u} which was on the correct side of the tangent candidate before the update to s_u can be on the wrong side of the tangent candidate after the update to s_u, and then it needs to be traversed again in order to be detected. The algorithm returns (s_0, s_1) in line 12 when it has traversed both polygons entirely with indices v_0 and v_1 after last updates to s_0 and s_1 without detecting any corner on the wrong side of the tangent candidate. That can happen only when $P_0, P_1 \subset \mathrm{RHP}(p_0[s_0], p_1[s_1])$. See Figure 4 for an example of how the algorithm proceeds.

In the test in line 5, $\Delta(a, b, c)$ denotes the filled triangle with corners a, b, c. If that test is positive, then $p_{1-u}[s_{1-u}]$ belongs to the convex hull of P_u, so $p_{1-u}[s_{1-u}] \neq r_{1-u}$. In that case, the boolean variable b_u is set, and then it prevents any updates to s_u in line 8 until it is cleared after a later update to s_{1-u} in line 8. It will be shown in the proof of Lemma 3 that such an update to s_{1-u} must occur if the convex hulls of P_0 and P_1 are not nested.

The main effort in proving correctness of Algorithm 1 lies in the following lemma, which is proved in Section 4.

▶ **Lemma 2.** *If the outer common tangents of P_0 and P_1 exist, then the loop in line 2 of Algorithm 1 ends with $s_0 < 2n_0$ and $s_1 < 2n_1$.*

The above implies that the algorithm ends up returning (s_0, s_1) in line 12 provided that $b_0 = b_1 =$ `false` when the loop in line 2 ends (this will be proved in Lemma 3).

To explain the role of the special variables b_0 and b_1, suppose temporarily that the conditions $s_0 < 2n_0$ and $s_1 < 2n_1$ are omitted from the test in line 2. If we were making the updates in line 8 regardless of the current values of b_0 and b_1, the algorithm could never end making updates to s_0 and s_1 even if the outer common tangents exist (see [2] for an example of such a behavior). In particular, Lemma 2 would no longer be true. On the other hand, if the convex hulls of P_0 and P_1 are nested, then one of the following happens:

- the algorithm never ends making updates to s_0 and s_1,
- one of b_0, b_1, say b_k, is `true` and the algorithm has traversed P_{1-k} entirely with the index v_{1-k} after last update to s_{1-k} without detecting any corner on the wrong side of the tangent candidate.

In both cases, taking the conditions $s_0 < 2n_0$ and $s_1 < 2n_1$ in line 2 back into account, the algorithm reports that the convex hulls of P_0 and P_1 are nested in line 11.

▶ **Lemma 3.** *If the outer common tangents of P_0 and P_1 exist, then the loop in line 2 of Algorithm 1 ends with $b_0 = b_1 =$* `false`*.*

Proof. We prove a slightly stronger statement, namely, that at most one of b_0 and b_1 can be `true` at a time, and if one of b_0 and b_1 is `true`, then it will be cleared subsequently. Hence, the algorithm cannot terminate with $b_0 =$ `true` or $b_1 =$ `true`.

Consider an iteration i of the loop in line 2 which leads to changing the value of b_0 from `false` to `true` in line 6. By induction, we can assume that $b_1 =$ `false`. Since the test in line 5 is positive, the edge $P_0[v_0 - 1, v_0]$ intersects $\mathcal{L}(p_0[s_0], p_1[s_1])$ at a point x such that $p_1[s_1]$ lies on the segment $p_0[s_0]x$. Moreover, $P_0[p_0[s_0], x] \subset \mathrm{RHP}(p_0[s_0], p_1[s_1])$, otherwise b_0 would be set before. Let y be the first corner of P_1 after $p_1[s_1]$ such that $y \notin \mathrm{RHP}(p_0[s_0], p_1[s_1])$. Such a corner exists, otherwise P_1 would be contained in the convex hull of P_0. It follows that the test in line 4 will be positive in the first iteration j after i in which $u = 1$ and $p_1[v_1] = y$. The edge $P_1[v_1 - 1, v_1]$ intersects $\mathcal{L}(p_0[s_0], p_1[s_1])$ at a point on the segment $p_0[s_0]x$, and hence the test in line 5 is negative in iteration j. Therefore, b_0 is cleared and we again have $b_0 = b_1 =$ `false`. The same argument shows that b_1 will be cleared after being set. ◀

▶ **Theorem 4.** *Algorithm 1 is correct, runs in linear time, and uses constant workspace. Specifically, if the outer common tangents exist, then Algorithm 1 returns a pair of indices (s_0, s_1) such that $(r_0, r_1) = (p_0[s_0], p_1[s_1])$, that is, $P_0, P_1 \subset \mathrm{RHP}(p_0[s_0], p_1[s_1])$. Otherwise, the algorithm returns* `nested`*.*

Proof. First, suppose the algorithm returns (s_0, s_1) in line 12. Consider the final values of s_0, s_1, b_0 and b_1. Due to the test in line 10, we have $s_0 < 2n_0$, $s_1 < 2n_1$, and $b_0 = b_1 =$ `false`, so the loop in line 2 has ended because $v_0 \geq s_0 + n_0$ and $v_1 \geq s_1 + n_1$. After the last update to (s_0, s_1), the test in line 4 has been performed for every $v_0 \in \{s_0 + 1, \ldots, s_0 + n_0\}$ and every $v_1 \in \{s_1 + 1, \ldots, s_1 + n_1\}$ and was negative – otherwise a further update would have been performed in line 8, as $b_0 = b_1 =$ `false`. This shows that $P_0, P_1 \subset \mathrm{RHP}(p_0[s_0], p_1[s_1])$.

Now, suppose that the outer common tangents exist. By Lemma 2 and Lemma 3, the loop in line 2 ends with $s_0 < 2n_0$, $s_1 < 2n_1$, and $b_0 = b_1 =$ `false`. Hence (s_0, s_1) is returned in line 12. In view of the discussion above, this proves correctness of the algorithm.

It is clear that Algorithm 1 uses constant workspace. For the running time, note that if an update to (s_0, s_1) happens in iteration i, the sum $s_0 + s_1$ is increased by at least $\frac{i-j}{2}$, where j is the number of the previous iteration in which an update to (s_0, s_1) happened or $j = 0$ if there has been no update before. By induction, we see that there have been at most $2(s_0 + s_1)$ iterations until an update to (s_0, s_1). Suppose first that $s_0 < 2n_0$ and $s_1 < 2n_1$ when the loop in line 2 terminates. There have been at most $4(n_0 + n_1)$ iterations until the final update to (s_0, s_1). Thereafter, at most $2\max\{n_0, n_1\} \leq 2(n_0 + n_1)$ iterations follow until $v_0 \geq s_0 + n_0$ and $v_1 \geq s_1 + n_1$, when the loop in line 2 terminates. Hence, there are at most $6(n_0 + n_1)$ iterations in total. Now, suppose that $s_0 \geq 2n_0$ or $s_1 \geq 2n_1$ when the loop terminates. By the same argument, the second to last update to (s_0, s_1) happens after at most $4(n_0 + n_1)$ iterations, after which at most $2(n_0 + n_1)$ iterations follow until the last update to (s_0, s_1). The loop is terminated immediately after the last update. Hence, we get

the same bound of $6(n_0 + n_1)$ iterations. Clearly, each iteration takes constant time, so the total running time of the algorithm is linear. ◀

4 Proof of Lemma 2

For our analysis, it will be convenient to imagine the execution of Algorithm 1 in continuous time. By considering various discrete events happening during the continuous execution of the algorithm, we are able to prove the invariant stated in Lemma 2.

4.1 Additional Terminology and Notation

For $U \subseteq \mathbb{R}^2$, let $\mathcal{F}(U)$ denote the set of compact subsets of U. By an *interval*, we mean a bounded interval of real numbers. We allow an interval to be closed or open at each endpoint independently. We shall consider functions defined on an interval I with the following sets (or their subsets) as codomains: $\mathbb{R}$ with the standard metric, $\mathbb{R}^2$ with the Euclidean metric, and $\mathcal{F}(\mathbb{R}^2)$ with the Hausdorff metric, a set $\mathcal{S}$ of functions with the discrete metric, and the power set $2^{\mathcal{S}}$ of a set $\mathcal{S}$ of functions, again with the discrete metric. The only purpose of these metrics is to have a suitable notion of convergence. We think of the domain I as *time*. If f is a function with domain I and I' is a subinterval of I, then $f \restriction I'$ denotes the restriction of f to I'. For a function $f\colon I \to X$, where X is (a subset of) one of the codomains above, a point in time $t \in I$ is a *discontinuity* of f if f is not continuous at t. We write

- $f(\nearrow t^\star)$ to denote the limit of $f(t)$ as $t \to t^\star$ from below, where $t^\star \in I \setminus \{\inf I\}$,
- $f(\searrow t^\star)$ to denote the limit of $f(t)$ as $t \to t^\star$ from above, where $t^\star \in I \setminus \{\sup I\}$.

If the limits $f(\nearrow t^\star)$ exist for all $t^\star \in I \setminus \{\inf I\}$ and the limits $f(\searrow t^\star)$ exist for all $t^\star \in I \setminus \{\sup I\}$, then we say that f has *one-sided limits*. Each of the functions f that we consider has one-sided limits and finitely many discontinuities. Note that f has a discontinuity at a point in time $t \in I$ if and only if $f(\nearrow t) \neq f(t)$ or $f(\searrow t) \neq f(t)$. A function $f\colon I \to \mathcal{F}(U)$, where $U \subseteq \mathbb{R}^2$, is *monotonically decreasing* if $f(t) \supseteq f(t')$ for any $t, t' \in I$ such that $t < t'$.

▶ **Lemma 5.** *Let I be an interval and $f\colon I \to \mathcal{F}(U)$ be a function with one-sided limits and finitely many discontinuities, where $U \subseteq \mathbb{R}^2$. Suppose $f \restriction I'$ is monotonically decreasing for every subinterval $I' \subseteq I$ such that $f \restriction I'$ is continuous on I'. Furthermore, suppose that*

- *$f(\nearrow t) \supseteq f(t)$ for any $t \in I \setminus \{\inf I\}$ such that $f(\nearrow t) \neq f(t)$,*
- *$f(t) \supseteq f(\searrow t)$ for any $t \in I \setminus \{\sup I\}$ such that $f(t) \neq f(\searrow t)$.*

Then f is monotonically decreasing in the entire domain I.

Proof. Let $t_1 < \cdots < t_n$ be the discontinuities of f. Let $t, t' \in I$ and $t < t'$. If there is no i with $t \leq t_i \leq t'$, then $f \restriction [t, t']$ is continuous, so it follows from the assumption that $f(t) \supseteq f(t')$. Otherwise, let i be minimum and j be maximum such that $t \leq t_i \leq t_j \leq t'$. If $t < t_i$, then the assumptions yield $f(t) \supseteq f(\nearrow t_i) \supseteq f(t_i)$, Similarly, the assumptions yield $f(t_k) \supseteq f(\searrow t_k) \supseteq f(\nearrow t_{k+1}) \supseteq f(t_{k+1})$ for $k \in \{i, \ldots, j-1\}$, and $f(t_j) \supseteq f(\searrow t_j) \supseteq f(t')$ if $t_j < t'$. Thus $f(t) \supseteq f(t')$. ◀

4.2 Continuous Interpretation of the Algorithm

Let m denote the number of iterations of the loop in line 2 performed by Algorithm 1. For $i \in \{0, \ldots, m\}$ and $k \in \{0, 1\}$, let $v_k(i)$ and $s_k(i)$ denote the values of v_k and s_k, respectively, after i iterations of the loop. In particular, $v_k(0) = s_k(0) = 0$. For $x \in \mathbb{R} \setminus \mathbb{Z}$, let $p_k[x]$ denote the interpolated point $(\lceil x \rceil - x)p_k[\lfloor x \rfloor] + (x - \lfloor x \rfloor)p_k[\lceil x \rceil]$ on the edge $P_k[\lfloor x \rfloor, \lceil x \rceil]$.

We extend the functions s_0 and s_1 to the real interval $[0, m]$ as follows. We imagine that the ith iteration of the loop in line 2 starts at time $i-1$ and ends at time i, and during that iteration v_u grows continuously from $v_u(i-1) = v_u(i) - 1$ to $v_u(i)$. Thus we define $v_u(t) = v_u(i) - i + t$ for $t \in (i-1, i)$. Suppose that the update in line 8 is to be performed in the ith iteration. If $s_u(i-1) = v_u(i-1)$, then all of the edge $P_u[v_u(i-1), v_u(i)]$ is in $\mathrm{LHP}(p_0[s_0(i-1)], p_1[s_1(i-1)])$. We therefore imagine that the update happens at time $i-1$ and then s_u grows continuously together with v_u up to $v_u(i)$; thus we define $s_u(t) = v_u(t)$ and $v_{1-u}(t) = s_{1-u}(i-1)$ for $t \in (i-1, i)$. If $s_u(i-1) < v_u(i-1)$, then the edge $P_u[v_u(i-1), v_u(i)]$ intersects the tangent candidate at a point $p_u[v_u(t^\star)]$, where $t^\star \in (i-1, i)$. We therefore imagine that the update in line 8 happens at time $t^\star$ and then s_u grows continuously together with v_u up to $v_u(i)$; thus we define

- $s_u(t) = s_u(i-1)$ and $v_{1-u}(t) = v_{1-u}(i-1)$ for $t \in (i-1, t^\star]$,
- $s_u(t) = v_u(t)$ and $v_{1-u}(t) = s_{1-u}(i-1)$ for $t \in (t^\star, i)$,

and we say that s_u *jumps* from $s_u(t^\star)$ to $v_u(t^\star) = s_u(\searrow t^\star)$ at time $t^\star$. Finally, in either case, we define $s_{1-u}(t) = s_{1-u}(i-1)$ for $t \in (i-1, i)$. The functions $s_0, s_1 \colon [0, m] \to \mathbb{R}$ thus defined are nondecreasing, have one-sided limits and finitely many discontinuities, and are left-continuous, that is, $s_0(\nearrow t) = s_0(t)$ and $s_1(\nearrow t) = s_1(t)$ for every $t \in (0, m]$. We have also defined functions $v_0, v_1 \colon [0, m] \to \mathbb{R}$, but we are not going to use them any more.

▶ **Observation 6.** *At any point in time during the execution of the continuous version of Algorithm 1, at most one of s_0, s_1 is changing. The tangent candidate $\mathcal{L}(p_0[s_0], p_1[s_1])$ either is not moving, or is turning continuously counterclockwise around $p_0[s_0]$ (when s_1 is changing), or is turning continuously clockwise around $p_1[s_1]$ (when s_0 is changing).*

The following is trivial if $s_k(t) = s_k(\searrow t)$ and otherwise is a direct consequence of the test in line 5 and of the fact that the update in line 8 is only performed when $b_u = \texttt{false}$.

▶ **Observation 7.** *If $t \in [0, m)$ and $k \in \{0, 1\}$, then $p_k[s_k(\searrow t)] \in \mathcal{R}(p_{1-k}[s_{1-k}(t)], p_k[s_k(t)])$ and $P_k[s_k(t), s_k(\searrow t)] \subset \mathrm{RHP}(p_0[s_0(t)], p_1[s_1(t)])$.*

4.3 Auxiliary Structure on the Polygons

In this subsection, we introduce some auxiliary concepts used in the proof of Lemma 2. They are defined in terms of the polygons P_0, P_1 only and are independent of the algorithm.

Assume for this entire subsection that the convex hulls of P_0 and P_1 are not nested. Thus there are two outer common tangents – let them be given by points $\ell_0, r_0 \in P_0$ and $\ell_1, r_1 \in P_1$ such that $P_0, P_1 \subset \mathrm{LHP}(\ell_0, \ell_1) \cap \mathrm{RHP}(r_0, r_1)$. Let $L = \ell_0\ell_1$ and $R = r_0r_1$. Let E be the polygonal region bounded by the chains $P_0[\ell_0, r_0]$, $P_1[\ell_1, r_1]$ and by the segments L, R. Since P_0 is oriented counterclockwise and P_1 clockwise, the interiors of P_0 and P_1 lie outside E.

▶ **Lemma 8.** *Every segment xy such that $xy \cap P_0 = \{x\}$ and $xy \cap P_1 = \{y\}$ is contained in E.*

Proof. The set $E \setminus (P_0[\ell_0, r_0] \cup P_1[\ell_1, r_1])$ separates P_0 and P_1 in $\mathrm{LHP}(\ell_0, \ell_1) \cap \mathrm{RHP}(r_0, r_1)$, so it contains a point z in common with the segment xy. If $z \in L$ or $z \in R$, then $xy = \ell_0\ell_1$ or $xy = r_0r_1$, respectively, so xy lies in E. So suppose z is in the interior of E. The segment zx cannot cross the boundary of E at any point other than x, and zy at any point other than y. This shows that xy lies in E. ◀

See Figure 5. Let $q_0 \in P_0$ and $q_1 \in P_1$ be fixed points such that at least one of q_0, q_1 is a corner of the respective polygon P_0 or P_1. Let $S = q_0q_1$. We consider the segment S as oriented from q_0 to q_1, so that we can speak of the *left side* of S, $\mathrm{LHP}(q_0, q_1)$, and the

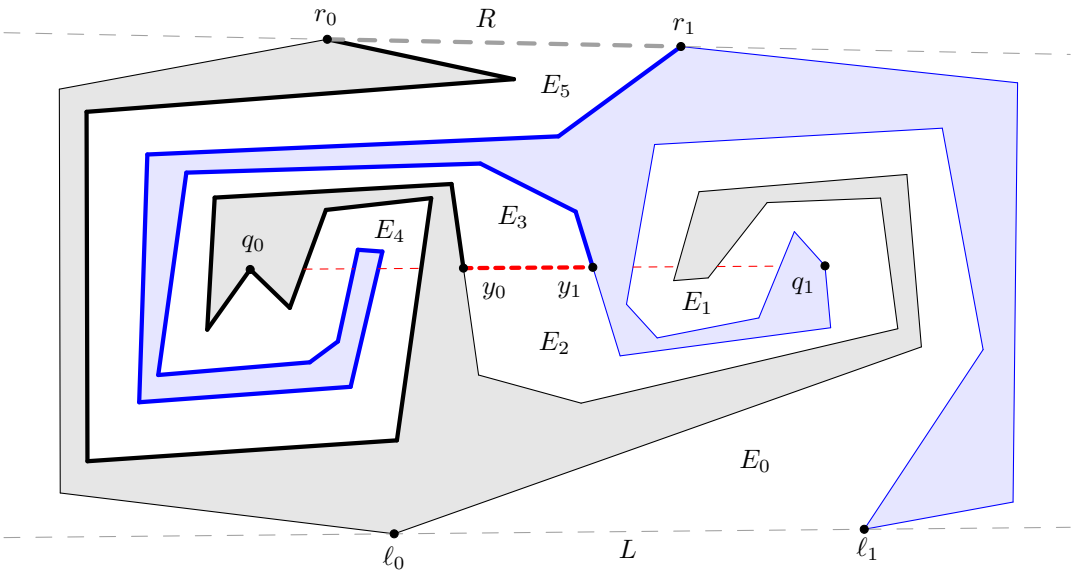

Figure 5 The doors are the five dashed segments on $S = q_0q_1$: D_4, D_5, D_3, D_1, D_2 in the order from q_0 to q_1. The weights of the doors are 2, 1, 1, −1, 0, respectively. $D_3 = y_0y_1$ is the primary door. The boundary of the primary region $E' = E_3 \cup E_4 \cup E_5$ is drawn with thick lines.

right side of S, $\mathrm{RHP}(q_0, q_1)$. A *door* is a subsegment xy of S such that $xy \cap P_k = \{x\}$ and $xy \cap P_{1-k} = \{y\}$ for some $k \in \{0, 1\}$. By Lemma 8, every door is contained in E. A *fence* is a subsegment xy of S such that $xy \subset E$, $xy \cap P_k = \{x, y\}$, and $xy \cap P_{1-k} = \emptyset$ for some $k \in \{0, 1\}$. Exceptionally, when S contains an edge xy of P_k, we call the whole edge xy a fence. Since at least one of q_0, q_1 is a corner, the latter is possible only when $x = q_k$ or $y = q_k$. Let $\mathcal{D}$ be the set of all doors defined by the fixed points q_0 and q_1. Figure 5 also illustrates the following lemma.

▶ **Lemma 9.** *The doors in $\mathcal{D}$ can be ordered as $D_1, \ldots, D_d$ so that if $D_i \cap P_0 = \{x_i\}$ and $D_i \cap P_1 = \{y_i\}$ for $i \in \{1, \ldots, d\}$, then*

- *the order of points along $P_0[\ell_0, r_0]$ is $\ell_0, x_1, \ldots, x_d, r_0$ (with possible coincidences),*
- *the order of points along $P_1[\ell_1, r_1]$ is $\ell_1, y_1, \ldots, y_d, r_1$ (with possible coincidences).*

The doors partition E into polygonal regions $E_0, \ldots, E_d$ such that

- *E_0 is bounded by L, $P_0[\ell_0, x_1]$, D_1 and $P_1[\ell_1, y_1]$ (it is degenerate when $D_1 = L$),*
- *E_i is bounded by D_i, $P_0[x_i, x_{i+1}]$, D_{i+1} and $P_1[y_i, y_{i+1}]$, for $i \in \{1, \ldots, d-1\}$,*
- *E_d is bounded by D_d, $P_0[x_d, r_0]$, R and $P_1[y_d, r_1]$ (it is degenerate when $D_d = R$).*

Proof. Suppose there are doors $xy, x'y' \in \mathcal{D}$ such that x is strictly before x' on $P_0[\ell_0, r_0]$ while y' is strictly before y on $P_1[\ell_1, r_1]$. It follows that the clockwise order of the four points along the boundary of E is x, x', y, y' and no two of these points coincide. By Lemma 8, both xy and $x'y'$ lie in E, so they must cross at a point different from their endpoints, which is a contradiction. This shows that the order of endpoints of the doors along $P_0[\ell_0, r_0]$ agrees with that along $P_1[\ell_1, r_1]$, which proves the first statement. The second statement is a straightforward corollary to the first. ◀

From now on, we use $D_1, \ldots, D_d$ to denote the doors in their order according to Lemma 9, and we use $E_0, \ldots, E_d$ to denote the regions defined in Lemma 9.

Recall that we consider S as a segment oriented from q_0 to q_1. Every door inherits that orientation, so that we can speak of the left side and the right side of the door. Taking into account that the regions E_{i-1} and E_i lie on opposite sides of D_i, we classify each door D_i as

- a *right-door* if E_{i-1} lies to the right and E_i lies to the left of D_i (in particular, if $D_i = L$),
- a *left-door* if E_{i-1} lies to the left and E_i lies to the right of D_i (in particular, if $D_i = R$).

▶ **Lemma 10.** *Consider a chain $P_k[a,b]$, where $k \in \{0,1\}$. If $P_k[a,b] \cap S = \{a,b\}$ and $P_k[a,b] \subset \mathrm{RHP}(q_0,q_1)$, then all doors contained in the segment ab occur in pairs of a left-door followed by a right-door, consecutive in the order on $\mathcal{D}$.*

Proof. Consider the polygonal region F bounded by the chain $P_k[a,b]$ and by the segment ab. It follows that $F \subset \mathrm{RHP}(q_0,q_1)$. Each of the regions $E_0,\dots,E_d$ lies either inside or outside F, where E_0 and E_d lie outside F. Each region E_i lying inside F connects the door D_i, which is therefore a left-door, and the door D_{i+1}, which is therefore a right-door. ◀

So far we were considering q_0 and q_1 as fixed points. Now, we allow them to change in time. Specifically, let I be a real interval that can be open or closed at each endpoint independently, and consider q_0 and q_1 as continuous functions $q_0\colon I \to P_0$ and $q_1\colon I \to P_1$. This way S becomes a continuous function $S\colon I \to \mathcal{F}(\mathbb{R}^2)$. Furthermore, suppose at least one of $q_0(t)$, $q_1(t)$ is a corner of the respective polygon for every $t \in I$, so that $S(t)$ can contain at most one other corner (by the general position assumption). Let $X(t)$ denote the set of intersection points of $S(t)$ with $P_0 \cup P_1$. In the exceptional case that $S(t)$ contains an edge of P_0 or P_1, we only include the endpoints of the edge in $X(t)$. The points in $X(t)$ are changing continuously except that an intersection point appears or disappears at a point in time $t \in I$ when $S(t)$ sweeps over a corner whose both incident edges lie on the same side of $S(t)$. Note that since the corners of P_0 and P_1 are assumed to be in general position and one of q_0 and q_1 is a corner, at most one point can appear in or disappear from $X(t)$ at any point in time. The doors are changing continuously except when one of the following *door events* happens as a point appears in or disappears from $X(t)$:

1. a fence splits into two doors,
2. two doors merge into a fence,
3. a door splits into a smaller door and a fence,
4. a door and a fence merge into a larger door.

Specifically, every door D can be represented as a continuous function $D\colon I_D \to \mathcal{F}(\mathbb{R}^2)$, where I_D is a subinterval of I (open or closed at each endpoint independently) such that

1. if $t = \inf I_D \in I_D$, then an endpoint of $D(t)$ is in $X(t)$ but not in $X(\nearrow t)$,
2. if $t = \sup I_D \in I_D$, then an endpoint of $D(t)$ is in $X(t)$ but not in $X(\searrow t)$,
3. if $t = \sup I_D \notin I_D$, then an interior point of $D(\nearrow t)$ is in $X(t)$ but not in $X(\nearrow t)$,
4. if $t = \inf I_D \notin I_D$, then an interior point of $D(\searrow t)$ is in $X(t)$ but not in $X(\searrow t)$.

At any point in time $t \in I$, the set of doors $\mathcal{D}(t)$ consists of the doors D such that $t \in I_D$ ordered according to Lemma 9. The following observation, a straightforward consequence of Lemma 9, summarizes how $\mathcal{D}(t)$ and the order on $\mathcal{D}(t)$ are changing in time.

▶ **Observation 11.** *The set $\mathcal{D}(t)$ and the order on $\mathcal{D}(t)$ are constant in time intervals where no door event happens. A door event at time t makes the following change to $\mathcal{D}(t)$:*

1. *if a fence splits into two doors D and D', then D and D' are added to $\mathcal{D}(\nearrow t)$ as consecutive doors to form $\mathcal{D}(t)$,*
2. *if two doors D and D' merge into a fence, then D and D' are consecutive in $\mathcal{D}(t)$ and they are removed from $\mathcal{D}(t)$ to form $\mathcal{D}(\searrow t)$,*

3. *if a door D splits into a smaller door D' and a fence, then D is replaced by D' in $\mathcal{D}(\nearrow t)$ to form $\mathcal{D}(t)$,*
4. *if a door D and a fence merge into a larger door D', then D is replaced by D' in $\mathcal{D}(t)$ to form $\mathcal{D}(\searrow t)$.*

In case of door events 1 and 2, the two doors D and D' are, in their order in $\mathcal{D}(t)$,

- *a right-door followed by a left-door if the edges incident to w lie to the right of $S(t)$,*
- *a left-door followed by a right-door if the edges incident to w lie to the left of $S(t)$,*

where w denotes the corner that triggers the event (i.e., the corner that appears in or disappears from $X(t)$ at time t). In case of door events 3 and 4, the door D' keeps the left/right-door status of D. The left/right-door status of every door D remains constant over the entire time interval I_D.

Now, consider q_0 and q_1 again as fixed points. Recall that $D_1, \ldots, D_d$ denote the doors in their order according to Lemma 9. We define the *weight* $W(D_i)$ of every door D_i by induction, as follows:

$$W(D_1) = \begin{cases} 1 & \text{if } D_1 \text{ is a right-door,} \\ -1 & \text{if } D_1 \text{ is a left-door,} \end{cases} \qquad W(D_i) = \begin{cases} W(D_{i-1}) + 1 & \text{if } D_i \text{ is a right-door,} \\ W(D_{i-1}) - 1 & \text{if } D_i \text{ is a left-door,} \end{cases}$$

for $i \in \{2, \ldots, d\}$. See Figure 5. The following is a direct consequence of Observation 11.

▶ **Observation 12.** *When $q_0 \colon I \to P_0$, $q_1 \colon I \to P_1$ are continuous functions, every door $D \colon I_D \to \mathcal{F}(\mathbb{R}^2)$ maintains constant weight over the entire time interval I_D. Furthermore, the function $W^\star \colon I \to \mathbb{Z}$ defined so that $W^\star(t)$ is the weight of the last door in the order on $\mathcal{D}(t)$ is constant over the entire time interval I.*

▶ **Lemma 13.** *For any fixed points q_0, q_1, there is at least one door with weight 1.*

Proof. The statement is obvious if $q_0 = \ell_0$ and $q_1 = \ell_1$, because in that case there is just one door L, which is a right-door by definition, so it has weight 1. To prove the lemma in general, let $I = [0, 1]$ and (abusing notation) consider arbitrary continuous functions $q_0 \colon I \to P_0$ and $q_1 \colon I \to P_1$ such that $q_0(0) = \ell_0$, $q_1(0) = \ell_1$, and $q_0(1)$, $q_1(1)$ are the points q_0, q_1 fixed in the statement of the lemma. By Observation 12, the function $W^\star \colon I \to \mathbb{Z}$ is constant over I, so $W^\star(1) = W^\star(0) = 1$ as observed above. This shows that the last door in the order on $\mathcal{D}(1)$ has weight 1. ◀

For any fixed points q_0, q_1, let the *primary door* D' be the first door with weight 1 in the order on $\mathcal{D}$. Such a door always exists due to Lemma 13.

▶ **Observation 14.** *The primary door D' is a right-door and is not preceded by a left-door in the order on $\mathcal{D}$.*

Let y_0 and y_1 denote the endpoints of D' so that $y_0 \in P_0$ and $y_1 \in P_1$. Let $Y_0 = P_0[y_0, r_0]$ and $Y_1 = P_1[y_1, r_1]$. Finally, let the *primary region* E' be defined as the polygonal region determined by D', Y_0, R and Y_1. See Figure 5.

▶ **Observation 15.** *If $D' = D_i$, then E' is the union of $E_i, \ldots, E_d$. In particular, E' contains the doors $D_{i+1}, \ldots, D_d$. By Observation 14, the region E' meets D' from the left.*

4.4 Back to the Algorithm

We recall the functions $s_0, s_1 \colon [0,m] \to \mathbb{R}$ describing the execution of Algorithm 1 as explained in Section 4.2, and we define functions $q_0 \colon [0,m] \to P_0$ and $q_1 \colon [0,m] \to P_1$ as follows:

$$q_0(t) = p_0[s_0(t)], \quad q_1(t) = p_1[s_1(t)] \quad \text{for } t \in [0,m].$$

They have the property that at least one of $q_0(t)$, $q_1(t)$ is a corner at any point in time $t \in [0,m]$. Some other objects that have been defined in Section 4.3 based on fixed points q_0, q_1 now become functions of time $t \in [0,m]$: the segment S, the primary door D', the points y_0, y_1, the chains Y_0, Y_1, and the primary region E'.

The functions q_0 and q_1 have finitely many discontinuities – the points of time $t \in [0,m)$ when the respective s_k jumps from $s_k(t)$ to $s_k(\searrow t)$. It is also clear that they have one-sided limits, since the functions s_0 and s_1 are bounded and piecewise monotone. It follows that the functions $D' \colon [0,m] \to \mathcal{F}(\mathbb{R}^2)$, $y_0 \colon [0,m] \to P_0$, $y_1 \colon [0,m] \to P_1$, $Y_0 \colon [0,m] \to \mathcal{F}(P_0)$, $Y_1 \colon [0,m] \to \mathcal{F}(P_1)$, and $E' \colon [0,m] \to \mathcal{F}(\mathbb{R}^2)$ also have one-sided limits and finitely many discontinuities, which arise from discontinuities of q_0, q_1 and from door events in between.

The following lemma is the heart of the proof of correctness of the algorithm. Informally speaking, it asserts that the primary region E' can only shrink in time, since the primary door D' always sweeps continuously into or jumps into E'.

▶ **Lemma 16.** *The functions Y_0 and Y_1 are monotonically decreasing.*

Proof. First, we let I be an arbitrary subinterval of $[0,m]$ in which q_0 and q_1 are continuous, and we prove the lemma for functions restricted to I: $y_0 \restriction I$, $y_1 \restriction I$, $Y_0 \restriction I$ and $Y_1 \restriction I$. Following the convention from Section 4.3, we consider doors as continuous functions $D \colon I_D \to \mathcal{F}(\mathbb{R}^2)$ with $I_D \subseteq I$ and, for $t \in I$, we let $\mathcal{D}(t)$ denote the set of doors D such that $t \in I_D$. Accordingly, we redefine $D'(t)$ to denote the function $D \colon I_D \to \mathcal{F}(\mathbb{R}^2)$ that is chosen as the primary door at time $t \in I$.

By Observation 12, every door $D \colon I_D \to \mathcal{F}(\mathbb{R}^2)$ maintains constant weight over the entire time interval I_D. By Observation 11, the only possible changes to $\mathcal{D}$ and to the order on $\mathcal{D}$ over time interval I are that doors are being added to or removed from $\mathcal{D}$. Therefore, any change to the choice of the primary door can only occur at a point in time $t \in I$ when a door event happens; moreover, the primary door $D'(t)$ must participate in that event, that is, if $D'(t) = D$, then $t = \inf I_D$ or $t = \sup I_D$.

Consider an interval $I' \subseteq I$ over which the choice of the primary door remains constant, that is, there is a door $D \colon I_D \to \mathcal{F}(\mathbb{R}^2)$ such that $I' \subseteq I_D$ and $D'(t) = D$ for every $t \in I'$. Since D is a continuous function, so are the functions $y_0 \restriction I'$, $y_1 \restriction I'$, $Y_0 \restriction I'$ and $Y_1 \restriction I'$. Furthermore, it follows from Observation 6 that the segment S is constant or is sweeping continuously to the left at any point in time $t \in I$. By Observation 15, D can only be moving towards the interior of E' in time interval I'. This shows that $E' \restriction I'$ and hence $Y_0 \restriction I'$ and $Y_1 \restriction I'$ are monotonically decreasing functions.

In view of Lemma 5, to complete the proof that $Y_0 \restriction I$ and $Y_1 \restriction I$ are monotonically decreasing, it remains to prove that

- $Y_0(\nearrow t) \supseteq Y_0(t)$ and $Y_1(\nearrow t) \supseteq Y_1(t)$ whenever $D'(\nearrow t) \neq D'(t)$, for $t \in I \setminus \{\inf I\}$,
- $Y_0(\searrow t) \subseteq Y_0(t)$ and $Y_1(\searrow t) \subseteq Y_1(t)$ whenever $D'(\searrow t) \neq D'(t)$, for $t \in I \setminus \{\sup I\}$.

We consider the kinds of door events as identified in Section 4.3, looking for events happening at time $t \in I$ that result in a primary door being added to or removed from $\mathcal{D}$.

1. A fence splits into two doors. Since $S(t)$ can only be sweeping to the left, both polygon edges incident to the corner triggering that event lie to the left of $S(t)$. Therefore, by

Observation 11, the two doors are a left-door followed by a right-door in the order on $\mathcal{D}(t)$. Consequently, by Observation 14, neither of the two doors can be primary.

2. Two doors merge into a fence. If $D'(t)$ is one of the two doors, then the choice of the primary door changes to some door $D \in \mathcal{D}(\searrow t) \subset \mathcal{D}(t)$ that is after $D'(t)$ in the order on $\mathcal{D}(t)$. By Lemma 9, the endpoints $y_0(\searrow t)$ and $y_1(\searrow t)$ of $D(t)$ lie on $Y_0(t)$ and $Y_1(t)$, respectively, so $Y_0(\searrow t) \subseteq Y_0(t)$ and $Y_1(\searrow t) \subseteq Y_1(t)$ as required.
3. A door splits into a smaller door and a fence. It follows from Observation 11 that the door added to $\mathcal{D}(t)$ maintains the weight of the door removed from $\mathcal{D}(\nearrow t)$. Therefore, assuming $D'(t) \neq D'(\nearrow t)$, $D'(t)$ is the door added to $\mathcal{D}(t)$ and $D'(\nearrow t)$ is the one removed from $\mathcal{D}(\nearrow t)$. Let $w \in P_k$ denote the corner that triggers the event, where $k \in \{0, 1\}$. It follows that $y_{1-k}(t) = y_{1-k}(\nearrow t)$, so $Y_{1-k}(t) = Y_{1-k}(\nearrow t)$. Since $w = y_k(t)$, we need to prove that $w \in Y_k(\nearrow t)$. Let $D = D'(\nearrow t)$ and let t_0 be a value in $I_D \cap I$ such that $t_0 < t$ and no door event happens in time interval $[t_0, t)$. For every $t' \in [t_0, t)$, let $\varphi(t')$ be the point on $D(t')$ closest to w. Since D moves continuously, φ is a continuous curve. Since $\varphi(t') \in E'(t')$ and $E'(t') \subset E'(t_0)$ as shown before for every $t' \in [t_0, t)$, φ must be contained in $E'(t_0)$. Since $w \in D(\nearrow t)$, we have $\varphi(\nearrow t) = w$. Furthermore, $E'(t_0)$ is a closed set, and hence $w \in E'(t_0)$. The interior of $E'(t_0)$ is disjoint from P_0 and P_1, so w must be a corner on the chain $Y_k(t_0) = P_k[y_k(t_0), r_k]$. Since $w = y_k(t)$, it follows that $Y_k(t) \subseteq Y_k(t_0)$. By letting t_0 approach t from below, we get $Y_k(t) \subseteq Y_k(\nearrow t)$.
4. A door and a fence merge into a larger door. Again, it follows from Observation 11 that the door added to $\mathcal{D}(\searrow t)$ maintains the weight of the door removed from $\mathcal{D}(t)$. Therefore, assuming $D'(t) \neq D'(\searrow t)$, $D'(t)$ is the door removed from $\mathcal{D}(t)$ and $D'(\searrow t)$ is the one added to $\mathcal{D}(\searrow t)$. Let $w \in P_k$ denote the corner that triggers the event, where $k \in \{0, 1\}$. It follows that $y_{1-k}(t) = y_{1-k}(\searrow t)$, so $Y_{1-k}(t) = Y_{1-k}(\searrow t)$. We make an argument similar to the one in the above case to show that $Y_k(\searrow t) \subseteq Y_k(t)$, but using reversed time. Let $D = D'(\searrow t)$ and let t_0 be a value in $I_D \cap I$ such that $t_0 > t$ and no door event happens in time interval $(t, t_0]$. For every $t' \in (t, t_0]$, let $\varphi(t')$ be the point on $D(t')$ closest to w. Since D moves continuously, φ is a continuous curve. For every $t' \in [0, m]$, let $F(t')$ be the polygonal region bounded by $P_0[\ell_0, y_0(t')]$, the primary door at time t', $P_1[\ell_1, y_1(t')]$, and the segment $L = \ell_0\ell_1$. Thus $F(t')$ is a sort of complementary region to $E'(t')$ in the region E. Since E' is monotonically decreasing on $(t, t_0]$ as shown before, F' is monotonically increasing on $(t, t_0]$. Therefore, since $\varphi(t') \in F(t')$, φ must be contained in $F(t_0)$. Since $w \in D(\searrow t)$, we have $\varphi(\searrow t) = w$. Furthermore, $F(t_0)$ is a closed set, and hence $w \in F(t_0)$. The interior of $F(t_0)$ is disjoint from P_0 and P_1, so w must be a corner on the chain $P_k[\ell_k, y_k(t_0)]$. Since $w = y_k(t)$, it follows that $P_k[\ell_k, y_k(t)] \subseteq P_k[\ell_k, y_k(t_0)]$. By letting t_0 approach t from above, we get $P_k[\ell_k, y_k(t)] \subseteq P_k[\ell_k, y_k(\searrow t)]$. Hence, $y_k(\searrow t)$ is on the chain $Y_k(t) = P_k[y_k(t), r_k]$ and therefore $Y_k(\searrow t) \subseteq Y_k(t)$.

Now, we return to the general case of functions y_0, y_1, Y_0 and Y_1 defined on the entire interval $[0, m]$. Consider a point in time $t \in [0, m)$ that is a discontinuity of q_k, where $k \in \{0, 1\}$. That is, s_k jumps from $s_k(t)$ to $s_k(\searrow t)$ at time t. We shall see that the jump of s_k has no effect on the choice of the primary door. By Observation 7, the point $p_k[s_k(\searrow t)] = q_k(\searrow t)$ lies on the ray $\mathcal{R}(q_{1-k}(t), q_k(t))$ and the chain $P_k[q_k(t), q_k(\searrow t)]$ belongs to $\mathrm{RHP}(q_0(t), q_1(t))$. Let $\mathcal{D}(t)$ and $\mathcal{D}(\searrow t)$ denote the sets of doors as defined for the segments $S(t)$ and $S(\searrow t)$, respectively. We shall prove that the primary door with respect to $\mathcal{D}(t)$ (i.e., defined for $S(t)$) is the same as with respect to $\mathcal{D}(\searrow t)$ (i.e., defined for $S(\searrow t)$).

Suppose $q_k(\searrow t)$ is on the segment $S(t)$. It follows that $S(\searrow t) \subset S(t)$, $\mathcal{D}(\searrow t) \subseteq \mathcal{D}(t)$, and $\mathcal{D}(t) \setminus \mathcal{D}(\searrow t)$ is the set of doors on $S(t) \setminus S(\searrow t)$. Since $P_k[q_k(t), q_k(\searrow t)] \subset \mathrm{RHP}(q_0(t), q_1(t))$, it follows from Lemma 10 that the doors in $\mathcal{D}(t) \setminus \mathcal{D}(\searrow t)$ occur in pairs of a left-door followed

by a right-door, consecutive in the order on $\mathcal{D}(t)$. Therefore, the weights of every door $D \in \mathcal{D}(\searrow t)$ with respect to the sets of doors $\mathcal{D}(\searrow t)$ and $\mathcal{D}(t)$ are equal. By Observation 14, none of the doors in $\mathcal{D}(t) \setminus \mathcal{D}(\searrow t)$ can be primary with respect to $\mathcal{D}(t)$, so the primary door is the same with respect to $\mathcal{D}(t)$ as with respect to $\mathcal{D}(\searrow t)$.

Now, suppose $q_k(\searrow t)$ is not on the segment $S(t)$. It follows that $S(t) \subset S(\searrow t)$, $\mathcal{D}(t) \subseteq \mathcal{D}(\searrow t)$, and $\mathcal{D}(\searrow t) \setminus \mathcal{D}(t)$ is the set of doors on $S(\searrow t) \setminus S(t)$. An argument analogous to that for $q_k(\searrow t) \in S(t)$ above shows that the primary door is the same with respect to $\mathcal{D}(\searrow t)$ as with respect to $\mathcal{D}(t)$.

To conclude, let $t_0 = 0$, $t_1, \ldots, t_{n-1}$ be the discontinuities of q_0 or q_1 ordered so that $t_1 < \cdots < t_{n-1}$, and $t_n = m$, and consider the closed intervals $I_i = [t_{i-1}, t_i]$ for $i \in \{1, \ldots, n\}$. Fix an index i and consider the restrictions $q_0 \restriction I_i$ and $q_1 \restriction I_i$. Only one of them, say $q_k \restriction I_i$, is not continuous, and the only discontinuity of $q_k \restriction I_i$ is t_{i-1}. As we have proved above, if we redefine $q_k(t_{i-1})$ by letting $q_k(t_{i-1}) = q_k(\searrow t_{i-1})$, the primary door $D'(t_{i-1})$ does not change, but then $q_k \restriction I_i$ becomes continuous. Therefore, what we have proved for restrictions of Y_0 and Y_1 to subintervals $I \subseteq [0, m]$ such that $q_0 \restriction I$ and $q_1 \restriction I$ are continuous implies that $Y_0 \restriction I_i$ and $Y_1 \restriction I_i$ are monotonically decreasing, for every $i \in \{1, \ldots, n\}$. The assumptions of Lemma 5 are satisfied for Y_0 and Y_1, so Y_0 and Y_1 are monotonically decreasing in the entire domain $[0, m]$. ◀

We are now ready to prove Lemma 2. Using the continuous interpretation of the algorithm, it can be rephrased as follows.

▶ **Lemma 17.** *For any $t \in [0, m]$, we have $0 \leq s_0(t) < 2n_0$ and $0 \leq s_1(t) < 2n_1$.*

Proof. We only present the proof of the bound on $s_0(t)$. That for $s_1(t)$ is analogous. Let $c_0(0)$ be the unique real in the interval $[0, n_0)$ such that $y_0(0) = p_0[c_0(0)]$. Let $\hat{c}_0$ be the unique real in the interval $[c_0(0), c_0(0) + n_0)$ such that $r_0 = p_0[\hat{c}_0]$. By Lemma 16, for $t \in (0, m]$, there is a unique real $c_0(t) \in [c_0(0), \hat{c}_0]$ such that $y_0(t) = p_0[c_0(t)]$, and this defines a nondecreasing function $c_0 \colon [0, m] \to \mathbb{R}$ with one-sided limits and finitely many discontinuities. Obviously, $0 \leq s_0(t)$ and $c_0(t) \leq \hat{c}_0 < c_0(0) + n_0 < 2n_0$. It remains to prove $s_0(t) \leq c_0(t)$ for $t \in [0, m]$.

We first prove that for every $t \in [0, m)$ with $s_0(t) \leq c_0(t)$, there is $\varepsilon > 0$ such that $s_0(t') \leq c_0(t')$ for all $t' \in [t, t + \varepsilon)$. Let $t \in [0, m)$ be such that $s_0(t) \leq c_0(t)$. First, suppose s_0 is continuous and either constant or strictly increasing on some interval $[t, t + \varepsilon)$ with $\varepsilon > 0$. If $s_0(t) < c_0(t)$, then the statement is clear, so suppose $s_0(t) = c_0(t)$. If s_0 is constant on $[t, t + \varepsilon)$, then the statement is clear, as c_0 is nondecreasing. If s_0 is strictly increasing on $[t, t + \varepsilon)$, we either have $c_0(t') = s_0(t')$ for $t' \in [t, t + \varepsilon')$, for some $\varepsilon' \in (0, \varepsilon]$, or c_0 jumps at time t to a higher value, that is, $c_0(t) < c_0(\searrow t)$. In both cases, the statement holds.

Now, suppose s_0 jumps at time t, that is, $s_0(t) < s_0(\searrow t)$. By choosing $\varepsilon > 0$ small enough, we can assume that s_0 is continuous on the interval $(t, t + \varepsilon)$ and that the points $\{p_0[s_0(t')] \colon t' \in (t, t + \varepsilon)\}$ are a part of one edge e of P_0, which also contains the point $p_0[s_0(\searrow t)]$. By Observation 7, we have $P_0[s_0(t), s_0(\searrow t)] \subset \mathrm{RHP}(p_0[s_0(t)], p_1[s_1])$. This and the facts that $p_0[c_0(t')] \in S(t')$ and $S(t') \cap \mathrm{RHP}(p_0[s_0(t)], p_1[s_1]) = \{p_1[s_1]\}$ imply $c_0(t') > s_0(\searrow t)$, for every $t' \in (t, t + \varepsilon)$. We conclude that for every $t' \in (t, t + \varepsilon)$, either $c_0(t') = s_0(t')$ or $p_0[c_0(t')]$ is on an edge of P_0 other than e, in which case $c_0(t') > s_0(t')$.

We now return to proving that $s_0(t) \leq c_0(t)$ for every $t \in [0, m]$. Suppose the contrary, and let $t^\star = \inf\{t \in [0, m] \colon s_0(t) > c_0(t)\}$. In view of the discussion above, we must have $s_0(t^\star) > c_0(t^\star)$. Then $t^\star > 0$, because $c_0(0) \geq 0 = s_0(0)$. By the definition of s_0, we have $s_0(\nearrow t^\star) = s_0(t^\star) > c_0(t^\star) \geq c_0(\nearrow t^\star)$. This contradicts the definition of $t^\star$. ◀

References

1 M. Abrahamsen. An optimal algorithm computing edge-to-edge visibility in a simple polygon. In *25th Canadian Conference on Computational Geometry (CCCG 2013)*, pages 157–162, 2013.

2 M. Abrahamsen. An optimal algorithm for the separating common tangents of two polygons. In *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *LIPIcs*, pages 198–208, 2015. arXiv:1511.04036 (corrected version).

3 T. Asano, K. Buchin, M. Buchin, M. Korman, W. Mulzer, G. Rote, and A. Schulz. Memory-constrained algorithms for simple polygons. *Comput. Geom.*, 46(8):959–969, 2013.

4 T. Asano, W. Mulzer, G. Rote, and Y. Wang. Constant-work-space algorithms for geometric problems. *J. Comput. Geom.*, 2(1):46–68, 2011.

5 L. Barba, M. Korman, S. Langerman, K. Sadakane, and R.I. Silveira. Space–time trade-offs for stack-based algorithms. *Algorithmica*, 72(4):1097–1129, 2015.

6 L. Barba, M. Korman, S. Langerman, and R.I. Silveira. Computing the visibility polygon using few variables. *Comput. Geom.*, 47(9):918–926, 2014.

7 G.S. Brodal and R. Jacob. Dynamic planar convex hull. In *43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pages 617–626, 2002.

8 E. Carson, J. Demmel, L. Grigori, N. Knight, P. Koanantakool, O. Schwartz, and H.V. Simhadri. Write-avoiding algorithms. Technical Report UCB/EECS-2015-163, University of California, Berkeley, 2015.

9 O. Darwish and A. Elmasry. Optimal time-space tradeoff for the 2D convex-hull problem. In *European Symposium on Algorithms (ESA 2014)*, volume 8737 of *LNCS*, pages 284–295. Springer, 2014.

10 L. Guibas, J. Hershberger, and J. Snoeyink. Compact interval trees: a data structure for convex hulls. *Int. J. Comput. Geom. Appl.*, 1(1):1–22, 1991.

11 S. Har-Peled. Shortest path in a polygon using sublinear space. In *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *LIPIcs*, pages 111–125, 2015.

12 J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *BIT Numer. Math.*, 32(2):249–267, 1992.

13 D. Kirkpatrick and J. Snoeyink. Computing common tangents without a separating line. In *4th International Workshop on Algorithms and Data Structures (WADS 1995)*, volume 955 of *LNCS*, pages 183–193. Springer, 1995.

14 M. Korman, W. Mulzer, A. van Renssen, M. Roeloffzen, P. Seiferth, and Y. Stein. Time-space trade-offs for triangulations and voronoi diagrams. In *Workshop on Algorithms and Data Structures (WADS 2015)*, volume 9214 of *LNCS*, pages 482–494. Springer, 2015.

15 A.A. Melkman. On-line construction of the convex hull of a simple polyline. *Inform. Process. Lett.*, 25(1):11–12, 1987.

16 M.H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. System Sci.*, 23(2):166–204, 1981.

17 F.P. Preparata and S.J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2):87–93, 1977.

18 G.T. Toussaint. Solving geometric problems with the rotating calipers. In *IEEE Mediterranean Electrotechnical Conference (MELECON 1983)*, pages A10.02/1–4, 1983.