On Romeo and Juliet Problems: Minimizing Distance-to-Sight

Hee-Kap Ahn
Department of Computer Science and Engineering, POSTECH
Pohang, South Korea
heekap@postech.ac.kr

Eunjin Oh
Department of Computer Science and Engineering, POSTECH
Pohang, South Korea
jin9082@postech.ac.kr

Lena Schlipf
Theoretische Informatik, FernUniversität in Hagen
Hagen, Germany
lena.schlipf@fernuni-hagen.de

Fabian Stehn
Institut für Informatik, Universität Bayreuth
Bayreuth, Germany
fabian.stehn@uni-bayreuth.de

Darren Strash
Department of Computer Science, Colgate University
Hamilton, USA
dstrash@cs.colgate.edu
https://orcid.org/0000-0001-7095-8749

Abstract

We introduce a variant of the watchman route problem, which we call the *quickest pair-visibility* problem. Given two persons standing at points s and t in a simple polygon P with no holes, we want to minimize the distance these persons travel in order to see each other in P. We solve two variants of this problem, one minimizing the longer distance the two persons travel (min-max) and one minimizing the total travel distance (min-sum), optimally in linear time. We also consider a query version of this problem for the min-max variant. We can preprocess a simple n-gon in linear time so that the minimum of the longer distance the two persons travel can be computed in $O(\log^2 n)$ time for any two query positions where the two persons lie.

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Paths and connectivity problems

Keywords and phrases Visibility polygon, shortest-path, watchman problems

Digital Object Identifier 10.4230/LIPIcs.SWAT.2018.6

Funding This work by Ahn and Oh was supported by the MSIT (Ministry of Science and ICT), Korea, under the SW Starlab support program (IITP-2017-0-00905) supervised by the IITP (Institute for Information & Communications Technology Promotion).

Acknowledgements This research was initiated at the 19th Korean Workshop on Computational Geometry in Würzburg, Germany.

1 Introduction

In the watchman route problem, a watchman takes a route to *guard* a given region—that is, any point in the region is visible from at least one point on the route. It is desirable to make the route as short as possible so that the entire area can be guarded as quickly as possible. The problem was first introduced in 1986 by Chin and Ntafos [4] and has been extensively studied in computational geometry [3, 14]. Though the problem is NP-hard for polygons with holes [4, 5, 7], an optimal route can be computed in time $O(n^3 \log n)$ for simple n-gons [6] when the tour must pass through a specified point, and $O(n^4 \log n)$ time otherwise.

In this paper, we study a variant we call the *quickest pair-visibility* problem, which can be stated as follows.

▶ **Problem** (quickest pair-visibility problem). *Given two points s and t in a simple polygon P, compute the minimum distance that s and t must travel in order to see each other in P.*

This problem may sound similar to the shortest path problem between s and t, in which the objective is to compute the shortest path for s to *reach* t. However, they differ even for a simple case: for any two points lying in a convex polygon, the distance in the quickest pair-visibility problem is zero while in the shortest path problem it is their Euclidean distance.

The quickest pair-visibility problem occurs in optimization tasks. For example, mobile robots that use a line-of-sight communication model are required to move to mutually-visible positions to establish communication [8]. An optimization task here is to find shortest paths for the robots to meet the visibility requirement for establishing communication among them.

Wynters and Mitchell [16] studied this problem for two agents acting in a polygonal domain in the presence of polygonal obstacles and gave an $O(nm)$-time algorithm for the min-sum variant (where m is the number of edges of the visibility graph of all corners) and an $O(n^3 \log n)$-time algorithm for the min-max variant. A query version of the quickest visibility problem has also been studied [1, 13, 15]. In the query problem, a polygon and a source point lying in the polygon are given, and the goal is to preprocess them and construct a data structure that allows, for a given query point, to find the shortest path taken from the source point to see the query point efficiently. Khosravi and Ghodsi [13] considered the case for a simple n-gon and presented an algorithm to construct a data structure of $O(n^2)$ space so that given a query, it finds the shortest visibility path in $O(\log n)$ time. Later, Arkin et al. [1] improved the result and presented an algorithm for the problem in a polygonal domain. Very recently, Wang [15] presented an improved algorithm for this problem for the case that the number of the holes in the polygon is relatively small. Figure 1(a) illustrates differences in these problems for a simple polygon and two points, s and t, in the polygon.

1.1 Our results

In this paper, we consider two variants of the quickest pair-visibility problem for a simple polygon: either we want to minimize the maximum length of a traveled path (*min-max variant*) or we want to minimize the sum of the lengths of both traveled paths (*min-sum variant*) We give a sweep-line-like approach that "rotates" the lines-of-sight along vertices on the shortest path between the start positions, allowing us to evaluate a linear number of candidate solutions on these lines. Throughout the sweep, we encounter solutions to both variants of the problem. We further show that our technique can be implemented in linear time.

We also consider a query version of this problem for the min-max variant. We can preprocess a simple n-gon in linear time so that the minimum of the longer distance the two query points travel can be computed in $O(\log^2 n)$ time for any two query points.

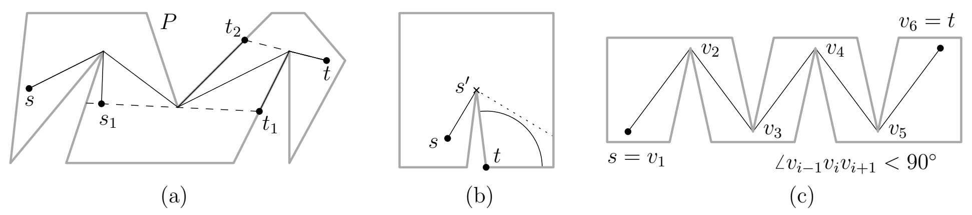

Figure 1 (a) The quickest pair-visibility problem finds two paths $\pi(s, s_1)$ and $\pi(t, t_1)$ such that $\overline{s_1t_1} \subset P$ and $\max\{|\pi(s, s_1)|, |\pi(t, t_1)|\}$ or $|\pi(s, s_1)| + |\pi(t, t_1)|$ is minimized. The quickest visibility problem for query point t finds a shortest $\pi(s, t_2)$ with $\overline{tt_2} \subset P$. (b) **min-max:** Every pair (s', t^*), where t^* is some point within the geodesic disk centered in t with radius $\pi(s, s')$, is an optimal solution to the *min-max* problem. (c) **min-sum:** Every pair (v_i, v_{i+1}) for $1 \le i < 6$ is an optimal solution to this instance.

2 Preliminaries

Let P be a simple polygon and ∂P be its boundary. The vertices of P are given in counter-clockwise order along ∂P. We denote the shortest path within P between two points $p, q \in P$ by $\pi(p, q)$ and its length by $|\pi(p, q)|$. Likewise, we denote the shortest path within P between a point $p \in P$ and a line segment $\ell \in P$ by $\pi(p, \ell)$. We say a point $p \in P$ is visible from another point $q \in P$ (and q is visible from p) if and only if line segment $\overline{pq}$ is completely contained in P.

For two starting points s and t, our task is to compute a pair (s', t') of points such that s' and t' are visible to each other, where we wish to minimize the lengths of $\pi(s, s')$, and $\pi(t, t')$. In the *min-max* setting, we wish to minimize $\max\{|\pi(s, s')|, |\pi(t, t')|\}$. For the *min-sum* setting, we wish to minimize $|\pi(s, s')| + |\pi(t, t')|$. Note that, for both variants, the optimum is not necessarily unique; see Figure 1(b) and (c).

For our discussion, let (s^*, t^*) be an optimal solution for the instance at hand. Let $V(p)$ denote the visible region for a point p in P, that is, the portion of P that is visible from p. Clearly, $V(p)$ is a *star-shaped* polygon. Moreover, every boundary edge of $V(p)$ is either (part of) an edge of P or a segment $\overline{vq}$ that is contained in P and parallel to $\overline{pv}$, where v is a vertex of P visible from p and q is a point on the boundary of P. We call an edge of the latter type a *window edge* of the visibility region. The structure of $V(p)$ may change as p moves along a path contained in P. It is known that a change to the structure of $V(p)$ occurs if and only if two vertices of P become collinear with p [2].

We say a segment g is tangent to a path π at a vertex v if $v \in g \cap \pi$ and v's neighboring vertices on π are on the same side of g.

▶ **Lemma 1.** *Unless s and t are visible to each other, the segment $\overline{s^*t^*}$ is tangent to the shortest path $\pi(s, t)$ at a vertex v of $\pi(s, t)$.*

Proof. We first show that there is a vertex of P lying on $\overline{s^*t^*}$. Consider the visibility regions $V(s^*)$ and $V(t^*)$. If s^* lies on a window edge e of $V(t^*)$, then e has a vertex v of P as its endpoint closer to t^*. Therefore v lies on $\overline{s^*t^*}$. The case that t^* lies on a window edge of $V(s^*)$ can be shown similarly. So assume that this is not the case, that is, s^* is in $V(t^*)$ but not in a window edge of $V(t^*)$ and t^* is in $V(s^*)$ but not in a window edge of $V(s^*)$. Then there is a point s' on $\pi(s, s^*) \cap V(t^*)$ infinitesimally close to s^* and a point t' on $\pi(t, t^*) \cap V(s^*)$ infinitesimally close to t^* such that $|\pi(s, s')| < |\pi(s, s^*)|$ and $|\pi(t, t')| < |\pi(t, t^*)|$, and s' and t' still see each other. This contradicts the optimality of (s^*, t^*).

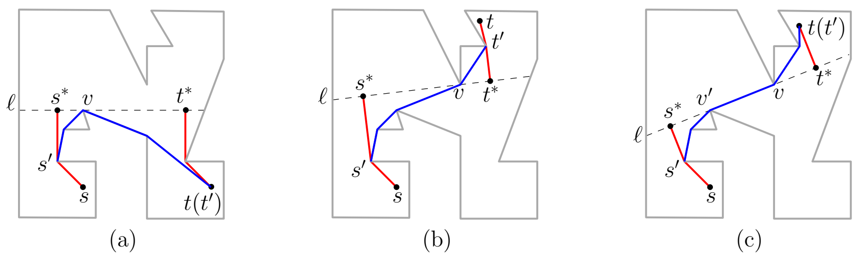

Figure 2 (a) Both s and t lie on the same side of ℓ through s^* and t^*. (b) If there is only one vertex v of P lying on $\overline{s^*t^*}$, we can always find another optimal pair of points by rotating ℓ around v and taking the closest points from s and t on the rotated ℓ under the geodesic metric. (c) The shortest path $\pi(s,t)$ passes through v and v'.

We now show that $\overline{s^*t^*}$ contains a vertex of $\pi(s,t)$. Let s' be the last vertex that $\pi(s,t)$ and $\pi(s,s^*)$ have in common from s, and let t' be the last vertex that $\pi(s,t)$ and $\pi(t,t^*)$ have in common from t. Since a subpath of a shortest path is also shortest, $\pi(s',t')$ is the subpath of $\pi(s,t)$ from s' to t'. Assume to the contrary that $\pi(s',t')$ (and therefore $\pi(s,t)$) contains no vertex of P that is also on $\overline{s^*t^*}$. There are two cases: (a) both s and t lie on the same side of the line ℓ through s^* and t^*, or (b) s and t lie on different sides of ℓ.

For case (a), ∂P touches ℓ at a vertex v lying between s^* and t^* locally from the same side of ℓ that s and t lie. Otherwise, (s^*,t^*) is not optimal as both $\pi(s,\ell)$ and $\pi(t,\ell)$ become shorter by a rotation of ℓ. See Figure 2(a). Consider the portion (line segment) of ℓ visible from v, which is split into two segments by v, one containing s^* and one containing t^*. Since $\pi(s',t')$ does not contain v, it has to cross the segment containing s^*. But then it must cross ℓ again in the segment containing t^* to reach t'. Since the portion of the path between the two crossing points can be shortened by the segment connecting them, which contradicts to the assumption that $\pi(s',t')$ is a shortest path. Thus v lies on $\pi(s,t)$ and $\overline{s^*t^*}$ is tangent to $\pi(s,t)$ at v.

For case (b), without loss of generality, assume that s lies below ℓ and t lies above ℓ. If there is only one vertex v of P lying on $\overline{s^*t^*}$, we can always find another pair of points $(\hat{s},\hat{t})$ such that $\hat{s}$ and $\hat{t}$ are visible to each other and they satisfy either (1) $|\pi(s,\hat{s})| < |\pi(s,s^*)|$ and $|\pi(t,\hat{t})| \leq |\pi(t,t^*)|$ or (2) $|\pi(s,\hat{s})| \leq |\pi(s,s^*)|$ and $|\pi(t,\hat{t})| < |\pi(t,t^*)|$. (The equality holds if s^* or t^* coincides with v.) Such points $\hat{s}$ and $\hat{t}$ can be obtained by rotating ℓ around v and taking the closest points from s and t, respectively, on the rotated ℓ under the geodesic metric. See Figure 2(b) for an illustration. Therefore we assume that there are two vertices v and v' that touch $\overline{s^*t^*}$ from above and from below, respectively. Since s lies below ℓ, v' comes before v from s^* along $\overline{s^*t^*}$ to t^*. See Figure 2(c). Consider the portion (line segment) of ℓ visible from v, which is split into two segments by v, one contains s^* and one contains t^*. If $\pi(s',t')$ crosses the segment containing s^*, it must cross ℓ again in the segment containing t^* to reach t'. Then the path between the two crossing points can be shortened by the segment connecting them, which is a contradiction. Thus, $\pi(s',t')$ passes through v'. The proofs for v and v' are symmetric, and thus both vertices v and v' are on the shortest path $\pi(s,t)$ which in turn also establishes, that $\overline{s^*t^*}$ is (locally) tangent to $\pi(s,t)$ at v and at v'. ◀

3 Computing All Events for a Sweep-Line-Like Approach

For each vertex v on $\pi(s,t)$ we compute a finite collection of lines through v, each being a configuration at which the combinatorial structure of the shortest paths $\pi(s,s^*)$ and/or $\pi(t,t^*)$ changes. To be more precise, at these lines either the vertices of $\pi(s,s^*)$ or $\pi(t,t^*)$

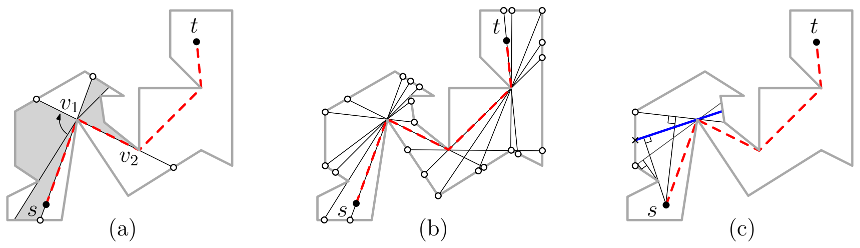

Figure 3 Path-, boundary-, and bend-events. (a) The endpoints of the line-of-sight through $\overline{sv_1}$ make up the first path-event. The line-of-sight rotates until it hits the next path-event: the endpoints of the line-of-sight through $\overline{v_1v_2}$. (b) All path- and boundary-events: the event-queue is initialized with these events. (c) A bend-event (marked with a cross) occurs between the two boundary-events. The shortest path from s to these segments changes at the bend-event.

(except for s^* and t^*) change or the edge of ∂P changes that is intersected by the extension of $\overline{s^*t^*}$. Notice that in the remaining part of the paper (s^*, t^*) is the optimal solution pair from s, t to the a given line (and not necessarily a global optimal solution for the quickest pair-visibility problem). To explain how to compute these lines, we introduce the concept of a *line-of-sight*.

▶ **Definition 2** (line-of-sight)**.** We call a segment ℓ a *line-of-sight* if (i) $\ell \subset P$, (ii) both endpoints of ℓ lie on ∂P, and (iii) ℓ is tangent to $\pi(s,t)$ at a vertex $v \in \pi(s,t)$.

The algorithm we present is in many aspects similar to a sweep-line strategy, except that we do not sweep over the scene in a standard fashion but rotate a *line-of-sight* ℓ in P around the vertices of the shortest path $\pi(s,t) := (s = v_0), v_1, \ldots, v_{k-1}, (t = v_k)$, making use of Lemma 1. The process will be initialized with a line-of-sight that contains s and v_1 and is then rotated around v_1 (while remaining tangent to v_1) until it hits v_2, see Figure 3(a). In general, the current line-of-sight is rotated around v_i in a way so that it remains tangent to v_i (it is rotated in the interior of P) until the line-of-sight contains v_i and v_{i+1}, then the process is iterated with v_{i+1} as the new rotation center. The process terminates as soon as the line-of-sight contains v_{k-1} and t.

While performing these rotations around the shortest path vertices, we encounter all combinatorially different lines-of-sight. As for a standard sweep-line approach, we will compute and consider events at which the structure of a solution changes: this is either because the interior vertices of $\pi(s, s^*)$ or $\pi(t, t^*)$ change or because the line-of-sight starts or ends at a different edge of ∂P. These events will be represented by points on ∂P (actually, we introduce the events as vertices on ∂P unless they are already vertices). Between two consecutive lines-of-sight, we compute the local minima of the relevant distances for the variant at hand in constant time and hence encounter all global minima eventually.

There are three event-types to distinguish:

1. **Path-Events** are endpoints of lines-of-sight that contain two consecutive vertices of the shortest path $\pi(s,t)$. See Figure 3(a).
2. **Boundary-Events** are endpoints of lines-of-sight that are tangent at a vertex of $\pi(s,t)$ and contain at least one vertex of $P \setminus \pi(s,t)$ (potentially as an endpoint). See Figure 3(b).
3. **Bend-Events** are endpoints of lines-of-sight where the shortest path from s (or t) to the line-of-sight gains or loses a vertex while rotating the line-of-sight around a vertex v. See Figure 3(c). Note that bend-events can coincide with path- or boundary-events.

We will need to explicitly know both endpoints of the line-of-sight on ∂P at each event and the corresponding vertex of $\pi(s,t)$ on which we rotate.

▶ **Lemma 3** (Computing path- and boundary-events). *For a simple polygon P with n vertices and points $s, t \in P$, the queue $\mathcal{Q}$ of all path- and boundary-events of the rotational sweep process, ordered according to the sequence in which the sweeping line-of-sight encounters them, can be initialized in $O(n)$ time.*

Proof. Consider some line-of-sight ℓ that is tangent to a vertex $v_i \in \pi(s,t)$ for some $0 < i < k$. Then ℓ subdivides P into a number of subpolygons. Consider ℓ as the union of two (sub)segments ℓ^+ and ℓ^- of ℓ induced by v_i such that $\ell^+ \cap \ell^- = \{v_i\}$ and ℓ^- is incident to the subpolygon of P induced by ℓ containing s.

We will discuss the computation of all boundary- and path-events swept by ℓ^+. The other events swept by ℓ^- can be computed in a second round by changing the roles of s and t. We do not maintain a queue for the events explicitly; instead we will introduce new vertices on ∂P or label existing vertices of ∂P as events. Later the events will be considered by following two pointers to vertices on ∂P and hence by processing the vertices in the order that they appear on ∂P.

We start with computing all path-events swept by ℓ^+. For this we compute the shortest path map M_s of s in P. The shortest path map of s is a decomposition of P in $O(n)$ triangular cells such that the shortest path from s to any point within a cell is combinatorially the same. It can be obtained by extending every edge of the shortest path tree of s towards its descendants until it reaches ∂P in linear time [10]. A path-event occurs when a line-of-sight contains two consecutive vertices of $\pi(s,t)$. Note that for each path-event, ℓ^+ appears as an edge of M_s and its endpoints appear as vertices of M_s. For each index i with $0 < i \leq k$, we find the edge incident to v_i and parallel to $\overline{v_{i-1}v_i}$ by considering every edge of M_s incident to v_i. This takes $O(n)$ time in total since there are $O(n)$ edges of M_s and we consider every edge at most once.

For computing the boundary-events, we use the following properties. While rotating around v_i from the position where ℓ contains v_{i-1} to the position in which ℓ contains v_{i+1}, let A_i^+ (A_i^-) be the region of P that is swept over by ℓ^+ (ℓ^-). (See Figure 4.) Observe that

P1 all A_i^+ for $1 < i < k$ are pairwise disjoint,
P2 all A_i^- for $1 < i < k$ are pairwise disjoint,
P3 for all $1 < i < k$ and all points $p \in A_i^+$ the shortest path $\pi(p,s)$ contains v_i,
P4 for all $1 < i < k$ and all points $p \in A_i^-$ the shortest path $\pi(p,t)$ contains v_i.

To compute all boundary-events that are vertices of P swept by ℓ^+, we will make use of the shortest path tree T_s for s in P. A boundary-event x is defined by a vertex $v_i \in \pi(s,t)$ such that the line-of-sight that contains x (potentially as one endpoint) is tangent to $\pi(s,t)$ in v_i. It follows from Property P3, that $\overline{v_i x}$ is an edge of T_s (and by that it cannot be obstructed by other edges of P) and $x \notin \pi(s,t)$. So the vertices of P whose parent vertex in T_s is a vertex of $\pi(s,t)$ are possible boundary-events. In order to compute all boundary-events we consider all consecutive path-events and compute all corresponding boundary-events by following ∂P and checking the vertices within the candidate set. We compute the boundary-events which are vertices of P swept by ℓ^- in a similar way.

So far we labeled all vertices x on ∂P that are boundary-events. We still need to compute the other endpoint $\tilde{x}$ of the line-of-sight $\overline{x\tilde{x}}$ that is tangent in v_i. Let $\overline{x_i\tilde{x}_i}$ be the line-of-sight at the path-event x_i so that $\tilde{x}_i, v_{i-1}, v_i, x_i \in \ell$. (See Figure 4.) While rotating ℓ around v_i, ℓ^+ sweeps over A_i^+ until the next path-event is met. Let E_i^+ be the sequence of the path- and boundary-events in A_i^+ we obtained so far sorted in counter-clockwise order along ∂P. The order of events in E_i^+ is the same as the order in which ℓ^+ sweeps over them. Our goal is to compute $\tilde{x}$ for every event in E_i^+ in order. To do this, we consider the (triangular) cells

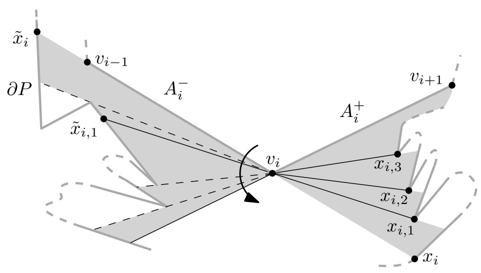

Figure 4 Let $E_i^+ = \langle x_{i,1}, \ldots, x_{i,k} \rangle$ for an index $1 \le k \le n$. We start at $\tilde{x}_i$ and follow the (triangular) cells of M_t incident to v_i in counter-clockwise order around v_i until we find $\tilde{x}_{i,1}$. Then we continue to follow such cells until we find $\tilde{x}_{i,2}$, and so on.

of M_t incident to v_i one by one in counter-clockwise order around v_i starting from the cell incident to $\tilde{x}_i$. Since every point in such cells is visible from v_i, we can determine if $\tilde{x}$ is contained in a cell in constant time for any event $x \in E_i^+$. Therefore, we can compute $\tilde{x}$ for every event x in E_i^+ in time linear in the number of the cells of M_t incident to v_i and the number of events of E_i^+, giving us all path- and boundary-events in $O(n)$ total time. ◀

Once we initialized the event queue $\mathcal{Q}$, we can now compute and process bend-events as we proceed in our line-of-sight rotations.

▶ **Lemma 4.** *All bend-events can be computed in $O(n)$ time, sorted in the order as they appear on the boundary of P.*

Proof. We assume that all path- and boundary-events are already computed. Additionally, we assume that all vertices of the boundary- and path-events (the endpoints of the corresponding line-of-sights) are inserted on ∂P. Recall that, for each event, we know both endpoints of the line-of-sight ℓ on ∂P and the corresponding vertex of $\pi(s,t)$ on which we rotate.

As in the proof of Lemma 3, we consider the line-of-sight ℓ tangent to a vertex $v \in \pi(s,t)$ as the union of two (sub)segments ℓ^+ and ℓ^- of ℓ induced by v such that $\ell^+ \cap \ell^- = \{v\}$ and ℓ^- is incident to the subpolygon of P induced by ℓ containing s. We discuss the computation of all bend-events that are encountered by ℓ^-. The bend-events that are swept over by ℓ^+ can be computed in a second round by changing the roles of s and t.

We start with the path-event defined by s and v_1, and consider all events in the order they appear. Let ℓ be the current line-of-sight rotating around a vertex v and denote by x the endpoint of ℓ^- other than v. To find the bend-events efficiently, we compute and maintain the shortest path $\pi(s,\ell)$ over the events.

While ℓ rotates around v, the combinatorial structure of $\pi(s,\ell)$ may change. Specifically, let $e_\ell = (u,w)$ denote the edge of $\pi(s,\ell)$ incident to ℓ with w on ℓ. Note that during the rotation of ℓ, all the edges of $\pi(s,\ell)$ are stationary, except that e_ℓ rotates around u. Therefore, a change in the combinatorial structure of $\pi(s,\ell)$ occurs only when (1) e_ℓ hits a vertex u' of P and splits into two edges sharing u' or (2) the two edges of $\pi(s,\ell)$ incident to u become parallel. (Then they merge into one and u disappears from the shortest path.) See Figure 5. From any event of the two event types above, e_ℓ, u, and $\pi(s,\ell)$ are updated accordingly. Additionally, x is updated and its new position is inserted as vertex on ∂P as it represents a bend-event.

▶ **Lemma 5.** *An event of type (1) occurs only when (a) x reaches a vertex u', or (b) e_ℓ hits a vertex u' of $\pi(s,t)$ in its interior. Moreover, for case (b), u and u' are consecutive in $\pi(s,t)$.*

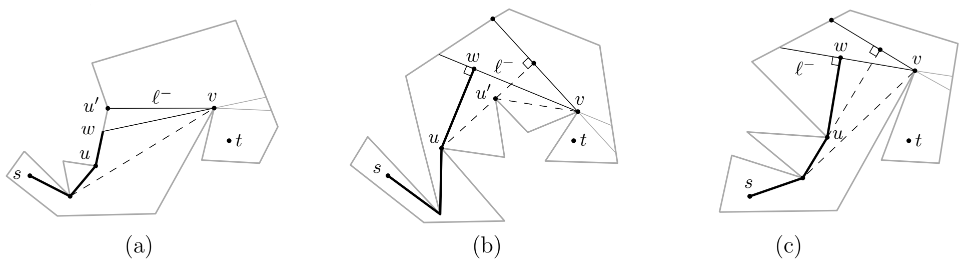

Figure 5 (a) A bend-event of type (1) occurs when $x = u_\ell$ reaches u'. (b) A bend-event of type (1) occurs when $e_\ell = \overline{uw}$ hits a vertex u' of $\pi(s,t)$. (c) A bend-event of type (2) occurs when two edges incident to u are parallel.

Proof. Consider the case that e_ℓ is not orthogonal to ℓ. Then the closest point in ℓ from s is x. Thus, the only way that e_ℓ hits a vertex of P is that x reaches u'. See Figure 5(a).

Now consider the case that e_ℓ is orthogonal to ℓ. Then u' is contained in $\pi(u,v)$. See Figure 5(b). Since $\pi(u,v)$ is a subpath of $\pi(s,t)$, u' is a vertex of $\pi(s,t)$, and thus u is the vertex of $\pi(s,t)$ previous to u' from s. ◀

▶ **Lemma 6.** *Once a vertex disappears from $\pi(s,\ell)$, it never appears again on the shortest path during the rotation of the current line-of-sight ℓ.*

Proof. Assume to the contrary that there is a vertex u that disappears from $\pi(s,\ell_1)$, but then appears again on $\pi(s,\ell_2)$ for two line-of-sights ℓ_1 and ℓ_2 during the rotation. Since both $\pi(s,\ell_1)$ and $\pi(s,\ell_2)$ contain u in its interior, both of them also contain $\pi(s,u)$. Since u disappears from $\pi(s,\ell_1)$, the edge of $\pi(s,\ell_1)$ incident to u is orthogonal to ℓ_1. We claim that u appears on $\pi(s,\ell_2)$ due to case (b) of type(1), that is, the edge of $\pi(s,\ell_2)$ incident to ℓ_2 hits u. Assume to the contrary that u appears on $\pi(s,\ell_2)$ due to case (a) of type (1). However, u (and its event vertex on ∂P) is already swept by a line-of-sight before we consider ℓ_2 because it appears on $\pi(s,\ell_1)$. Thus, u appears on $\pi(s,\ell_2)$ due to case (b) of (2), and the edge of $\pi(s,\ell_2)$ incident to u is orthogonal to ℓ_2. This means that ℓ_1 and ℓ_2 are parallel.

Since ℓ_1 and ℓ_2 are parallel, they are tangent to $\pi(s,t)$ at two distinct vertices, say v_1 and v_2, respectively. Moreover, the path $\pi(p_1,p_2)$ contains v_1 for any two points $p_1 \in P_1$ and $p_2 \in \ell_2$, where P_1 is the subpolygon bounded by ℓ_1^- containing s. Thus, $\pi(s,\ell_2)$ contains $\pi(s,v_1)$, and no vertex in P_1 other than the vertices of $\pi(s,v_1)$ appears on $\pi(s,\ell_2)$. Since u is contained in P_1, it cannot appear on $\pi(s,\ell_2)$, which is a contradiction. ◀

We can update u, e_ℓ, x and $\pi(s,\ell)$ in constant time for a type (1) event. We can update them in $O(n)$ time for all type (2) events in total by Lemma 6. The vertices representing the bend-events can be inserted on ∂P in the same time. ◀

4 Algorithm Based on a Sweep-Line-Like Approach

In this section, we present a linear-time algorithm for computing the minimum distance that two points s and t in a simple polygon P travel in order to see each order. We compute all events defined in Section 3 in linear time. The remaining task is to handle the lines-of-sight lying between two consecutive events.

▶ **Lemma 7.** *For any two consecutive events, the line-of-sight ℓ lying between them that minimizes the sum of the distances from s and t to ℓ can be found in constant time.*

Proof. Let $\mathcal{L}$ be the set of all lines-of-sight lying between the two consecutive events. Every line-of-sight in $\mathcal{L}$ contains a common vertex v of $\pi(s,t)$. We assume that $\mathcal{L}$ contains no vertical line-of-sight. Otherwise, we consider the set containing all lines-of-sight of $\mathcal{L}$ with positive slopes, and then the set containing all lines-of-sight of $\mathcal{L}$ with negative slopes.

By construction, the second to the last vertex u of $\pi(s,\ell)$ (and $\pi(t,\ell)$) for any $\ell \in \mathcal{L}$ remains the same. We already obtained v and u while computing the events. We will give an algebraic function for the length of $\pi(s,\ell)$ for $\ell \in \mathcal{L}$. An algebraic function for the length of $\pi(t,\ell)$ can be obtained by changing the roles of s and t.

Since the topology of $\pi(s,\ell)$ for every $\ell \in \mathcal{L}$ remains the same, we consider only the length of $\pi(u,\ell)$. Observe that $\pi(u,\ell)$ is a line segment for any $\ell \in \mathcal{L}$, and thus its length is the same as the Euclidean distance between u and ℓ. The length is either the Euclidean distance between u and the line containing ℓ, or the Euclidean distance between u and the endpoint of ℓ closest to u. We show how to handle the first case only because the second case can be handled analogously.

To use this observation, we use $\ell(\alpha)$ to denote the line of slope α passing through v for any $\alpha > 0$. There is an interval I such that $\ell(\alpha)$ contains a line-of-sight in $\mathcal{L}$ if and only if $\alpha \in I$. The Euclidean distance between u and $\ell(\alpha)$ is the same as the distance between u and the line-of-sight contained in $\ell(\alpha)$. Thus, in the following, we consider the distance between u and $\ell(\alpha)$ for every $\alpha \in I$.

Since $\ell(\alpha)$ passes through a common vertex, the line $\ell(\alpha)$ can be represented as the form of $y = \alpha x + f(\alpha)$, where $f(\alpha)$ is a function linear in α. Then, the distance between u and $\ell(\alpha)$ can be represented as the form of $|c_1\alpha + c_2|/\sqrt{\alpha^2+1}$, where c_1 and c_2 are constants depending only on v and u.

Then our problem reduces to the problem of finding a minimum of the function of the form of $(|c_1\alpha + c_2| + |c_1'\alpha + c_2'|)/\sqrt{\alpha^2+1}$ for four constants c_1, c_2, c_1' and c_2', and for all $\alpha \in I$. We can find a minimum in constant time using an elementary analysis. ◀

▶ **Lemma 8.** *For any two consecutive events, the line-of-sight ℓ lying between the them that minimizes the maximum of the distances from s and t to ℓ can be found in constant time.*

▶ **Theorem 9.** *Given a simple n-gon P with no holes and two points $s, t \in P$, a point-pair (s^*, t^*) such that* i) $\overline{s^*t^*} \subset P$ *and* ii) *either* $|\pi(s,s^*)| + \pi(t,t^*)|$ *or* $\max\{|\pi(s,s^*)|, |\pi(t,t^*)|\}$ *is minimized can be computed in $O(n)$ time.*

Proof. Our algorithm first computes all path- and boundary-events as described in Lemma 3. The number of events introduced during this phase is bounded by the number of vertices of the shortest path maps, M_s and M_t, respectively, which are $O(n)$. In the next step, it computes the bend-events on ∂P as described in Lemma 4, which can be done in $O(n)$ time. Finally, our algorithm traverses the sequence of events. Between any two consecutive events, it computes the respective local optimum in constant time by Lemma 7. It maintains the smallest one among the local optima computed so far, and return it once all events are processed. Therefore the running time of the algorithm is $O(n)$.

For the correctness, consider the combinatorial structure of a solution and how it changes. The path-events ensure that all vertices of $\pi(s,t)$ are considered as being the vertex lying on the segment connecting the solution (s^*, t^*). While the line-of-sight rotates around one fixed vertex of $\pi(s,t)$, either the endpoints of line-of-sight sweep over or become tangent to a vertex of ∂P. These are exactly the boundary-events. Or the combinatorial structure of $\pi(s,s^*)$ or $\pi(t,t^*)$ changes as interior vertices of $\pi(s,s^*)$ or $\pi(t,t^*)$ appear or disappear. These happen exactly at bend-events. Therefore, our algorithm returns an optimal point-pair. ◀

5 Quickest Pair-Visibility Query Problem

In this section, we consider a query version of the min-max variant of the quickest pair-visibility problem: Preprocess a simple n-gon P so that the minimum traveling distance for two query points s and t to see each other can be computed efficiently. We can preprocess a simple n-gon in linear time and answer a query in $O(\log^2 n)$ time by combining the approach in Section 4 with the data structure given by Guibas and Hershberger [9, 11]. For any two query points s and t in P, the query algorithm for their data structure returns $\pi(s,t)$ represented as a binary tree of height $O(\log n)$ in $O(\log n)$ time [11]. Thus, we can apply binary search on the vertices (or the edges) on $\pi(s,t)$ efficiently.

Imagine that we rotate a line-of-sight along the vertices of $\pi(s,t)$ for two query points s and t in P. Lemma 1 implies that there is a line-of-sight containing s^* and t^*, where (s^*,t^*) is an optimal solution. We call it an *optimal line-of-sight*. We define the order of any two lines-of-sight as the order in which they appear during this rotational sweep process. By the following lemma, we can apply binary search on the sequence of events along ∂P and find two consecutive events such that the respective local optimum achieved between them is a global optimal solution.

▶ **Lemma 10.** *The geodesic distance between s (and t) and the rotating line-of-sight increases (and decreases) monotonically as the line-of-sight rotates along the vertices of $\pi(s,t)$ from s.*

Proof. Let ℓ be a line-of-sight which is tangent to $\pi(s,t)$ at a vertex v. Consider the subdivision of P induced by ℓ and let P_s be the subpolygon that contains s. Let ℓ' be a line-of-sight that comes after ℓ during the rotational sweep process. We claim that ℓ' does not intersect the interior of P_s. If ℓ' is tangent to $\pi(s,t)$ at v, it never intersects the interior of P_s as shown in the proof of Lemma 3. Assume that ℓ' is tangent to $\pi(s,t)$ at a vertex u that comes after v along $\pi(s,t)$ from s, but intersects the interior of P_s. Without loss of generality, assume that ℓ is horizontal and P_s lies locally below ℓ. Then u must lie strictly above the line containing ℓ. However, since both v and u are vertices of $\pi(s,t)$ and ℓ is tangent to $\pi(s,t)$ at v, there must be another vertex u' of $\pi(s,t)$ that lies on or below the line containing ℓ and appears between v and u along $\pi(s,t)$. Thus, u is not visible from any point on ℓ, and ℓ' does not intersect the interior of P_s. Since $\pi(s,\ell')$ intersects ℓ, we have $\pi(s,\ell') \geq \pi(s,\ell)$. The claim for t and the rotating line-of-sight can be shown analogously. ◀

5.1 Binary Search for the Path-Events

We first consider the path-events, and find two consecutive path-events containing an optimal line-of-sight between them. Let $\pi(s,t) := (s = v_0), v_1, \ldots, v_{k-1}, (t = v_k)$. Due to the shortest-path data structure by Guibas and Hershberger, we can obtain $\pi(s,t)$ represented as a binary tree of height $O(\log n)$ in $O(\log n)$ time. Consider an edge $\overline{v_i v_{i+1}}$ of $\pi(s,t)$. We can determine whether or not an optimal line-of-sight is tangent to $\pi(s,t)$ at a vertex lying after v_i along $\pi(s,t)$ in $O(\log n)$ time. To do this, we compute the line-of-sight ℓ containing $\overline{v_i v_{i+1}}$ in $O(\log n)$ time [12] and compute the length of $\pi(s,\ell)$ and $\pi(t,\ell)$ in $O(\log n)$ time [9]. An optimal line-of-sight is tangent to $\pi(s,t)$ at a vertex lying after v_i if and only if $\pi(s,\ell)$ is shorter than $\pi(t,\ell)$. Therefore, we can compute the two consecutive path-events with an optimal solution lying between them in $O(\log^2 n)$ time.

5.2 Binary Search for the Boundary-Events

Now we have the vertex v_i of $\pi(s,t)$ contained in an optimal line-of-sight. We find two consecutive boundary-events defined by lines-of-sight tangent to $\pi(s,t)$ at v_i such that an optimal line-of-sight lies between them. Let $\tilde{x}_i$ and x_i be the first points of ∂P hit by the rays from any point in $\overline{v_{i-1}v_i}$ towards v_{i-1} and v_i, respectively. See Figure 4. Similarly, let $\tilde{x}_{i+1}$ and x_{i+1} be the first points of ∂P hit by the rays from any point in $\overline{v_i v_{i+1}}$ towards v_i and v_{i+1}, respectively. These four points of ∂P can be found in $O(\log n)$ time by the ray-shooting data structure [12]. Without loss of generality, we assume that a line-of-sight rotates around v_i in the counter-clockwise direction in the rotational sweep process. Let $\tilde{\gamma}$ be the part of ∂P lying from $\tilde{x}_i$ to $\tilde{x}_{i+1}$ in counter-clockwise order, and γ be the part of ∂P lying from x_i to x_{i+1} in counter-clockwise order. An optimal line-of-sight ℓ^* has one endpoint on $\tilde{\gamma}$ and the other endpoint on γ.

We first find the edge of $\tilde{\gamma}$ (resp. γ) containing an endpoint of ℓ^* by applying binary search on the vertices of $\tilde{\gamma}$ (resp. γ). This gives two consecutive boundary-events such that ℓ^* lies between them. We now show how to find the edge of γ containing an endpoint of ℓ^*. The edge on $\tilde{\gamma}$ can be found analogously.

We perform a binary search on the vertices in γ as follows. Let x^* be the endpoint of ℓ^* contained in γ. For any vertex u of γ, we can determine which part of γ with respect to u contains x^* in $O(\log n)$ time. To do this, we consider the line-of-sight ℓ containing the edge of $\pi(v_i, u)$ incident to v_i. Observe that ℓ intersects $\pi(v_i, u)$ only in the edge including its endpoints as $\pi(v_i, u)$ is a shortest path. See Figure 6(a). Since we can obtain the edge of $\pi(v_i, u)$ incident to v_i in $O(\log n)$ time using the shortest-path data structure, we can obtain ℓ in the same time. Here, to obtain the endpoint of ℓ on γ, we use the ray-shooting data structure that supports $O(\log n)$ query time [12]. Then we compare $d(s, \ell)$ and $d(t, \ell)$ in $O(\log n)$ time. The point x^* comes after u from x_i if and only if $d(s, \ell) < d(t, \ell)$. Therefore, we can determine which part of γ with respect to u contains x^* in $O(\log n)$ time, and thus the binary search is completed in $O(\log^2 n)$ time. In this way, we can compute two consecutive boundary-events such that an optimal line-of-sight lies between them in $O(\log^2 n)$ time.

5.3 Binary Search for the Bend-Events

Now we have two consecutive events in the sequence of all path- and boundary-events that contain an optimal line-of-sight ℓ^* between them. Let ℓ_1 and ℓ_2 be two lines-of-sight corresponding to the two consecutive events such that ℓ_2 comes after ℓ_1. The remaining task is to handle the bend-events lying between them. For the bend-events, we perform a binary search on the edges of $\pi(s, \ell_1) \cup \pi(s, \ell_2)$ in $O(\log^2 n)$ time. Then we perform binary search on the edges of $\pi(t, \ell_1) \cup \pi(t, \ell_2)$ in $O(\log^2 n)$ time. In the following, we describe the binary search on $\pi(s, \ell_1) \cup \pi(s, \ell_2)$. The other one can be done analogously.

We find the point s' such that $\pi(s, s')$ is the maximal common subpath of $\pi(s, \ell_1)$ and $\pi(s, \ell_2)$ from s in $O(\log n)$ time using the shortest-path data structure [11]. See Figure 6(b). Then we obtain $\pi' = \pi(s', \ell_1) \cup \pi(s', \ell_2)$ represented as a binary tree of height $O(\log n)$ in $O(\log n)$ time. For an edge e of π', we use $\ell(e)$ to denote the line-of-sight containing v_i and orthogonal to the line containing e. Observe that $\ell(e)$ comes after $\ell(e')$ if and only if e comes after e' along π' from ℓ_1. Also, given an edge e of π', we can compute $\ell(e)$ in constant time. Using these properties, we can find two consecutive edges e and e' of π' such that ℓ^* lies between $\ell(e)$ and $\ell(e')$ in $O(\log^2 n)$ time by applying binary search on π' as we did for path- and boundary-events.

Now we have two consecutive events in the sequence of all path-, boundary- and bend-events that contains ℓ^* between them. Recall that the combinatorial structure of $\pi(s, \ell)$ (and $\pi(t, \ell)$) is the same for every lines-of-sight lying between the two events. Let (u_s, w_s) and (u_t, w_t) be the edges of $\pi(s, \ell)$ and $\pi(t, \ell)$ incident to ℓ at w_s and w_t, respectively, for any line-of-sight ℓ lying between the two events. Using the shortest-path data structure, we can obtain $u_s, u_t, d(s, u_s)$ and $d(t, u_t)$ in $O(\log n)$ time. Then we apply the algorithm in

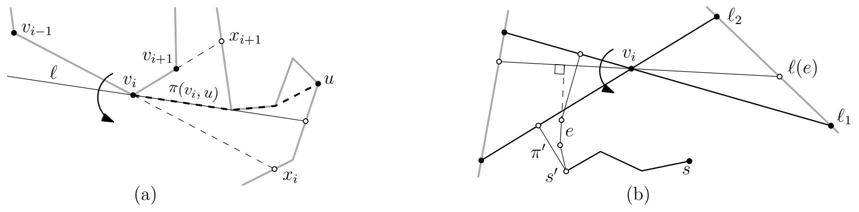

Figure 6 (a) The line-of-sight intersecting $\pi(v_i, u)$ contains the edge of $\pi(v_i, u)$ incident to v_i. (b) The maximal common subpath of $\pi(s, \ell_1)$ and $\pi(s, \ell_2)$ from s is $\pi(s, s')$.

Lemma 7 to find an optimal line-of-sight in constant time. In this way, we can obtain an optimal line-of-sight in $O(\log^2 n)$ time in total.

Therefore, we can find two consecutive events with an optimal solution between them, and we can obtain an optimal solution in $O(\log^2 n)$ time in total.

▶ **Theorem 11.** *Given a simple n-gon P, we can preprocess it in $O(n)$ time to find the minimum of the longer distance that s and t travel in order to see each other in P can be computed in $O(\log^2 n)$ time for any two query points $s, t \in P$.*

References

1 Esther M. Arkin, Alon Efrat, Christian Knauer, Joseph S. B. Mitchell, Valentin Polishchuk, Günter Rote, Lena Schlipf, and Topi Talvitie. Shortest path to a segment and quickest visibility queries. *Journal of Computational Geometry*, 7(2):77–100, 2016. `doi:10.20382/jocg.v7i2a5`.

2 Boris Aronov, Leonidas J. Guibas, Marek Teichmann, and Li Zhang. Visibility queries and maintenance in simple polygons. *Discrete & Computational Geometry*, 27(4):461–483, 2002. `doi:10.1007/s00454-001-0089-9`.

3 Svante Carlsson, Håkan Jonsson, and Bengt J. Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete & Computational Geometry*, 22(3):377–402, 1999. `doi:10.1007/PL00009467`.

4 Wei-pang Chin and Simeon C. Ntafos. Optimum watchman routes. In Alok Aggarwal, editor, *Proceedings of the Second Annual ACM SIGACT/SIGGRAPH Symposium on Computational Geometry, Yorktown Heights, NY, USA, June 2-4, 1986*, pages 24–33. ACM, 1986. `doi:10.1145/10515.10518`.

5 Wei-pang Chin and Simeon C. Ntafos. Optimum watchman routes. *Inf. Process. Lett.*, 28(1):39–44, 1988. `doi:10.1016/0020-0190(88)90141-X`.

6 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 473–482. ACM, 2003. `doi:10.1145/780542.780612`.

7 Adrian Dumitrescu and Csaba D. Tóth. Watchman tours for polygons with holes. *Comput. Geom.*, 45(7):326–333, 2012. `doi:10.1016/j.comgeo.2012.02.001`.

8 Anurag Ganguli, Jorge Cortes, and Francesco Bullo. Visibility-based multi-agent deployment in orthogonal environments. In *Proceedings of the 2007 American Control Conference (ACC '07)*, pages 3426–3431, 2007. `doi:10.1109/ACC.2007.4283034`.

9 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39(2):126–152, 1989. `doi:10.1016/0022-0000(89)90041-X`.

10 Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert Endre Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987. `doi:10.1007/BF01840360`.

11 John Hershberger. A new data structure for shortest path queries in a simple polygon. *Inf. Process. Lett.*, 38(5):231–235, 1991. doi:10.1016/0020-0190(91)90064-O.

12 John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18(3):403–431, 1995. doi:10.1006/jagm.1995.1017.

13 Ramtin Khosravi and Mohammad Ghodsi. The fastest way to view a query point in simple polygons. In *Proceedings of the 21st European Workshop on Computational Geometry*, pages 187–190, 2005.

14 Joseph S. B. Mitchell. Approximating watchman routes. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 844–855. SIAM, 2013. doi:10.1137/1.9781611973105.60.

15 Haitao Wang. Quickest visibility queries in polygonal domains. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, volume 77 of *LIPIcs*, pages 61:1–61:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.SoCG.2017.61.

16 Erik L. Wynters and Joseph S. B. Mitchell. Shortest paths for a two-robot rendez-vous. In *Proceedings of the 5th Canadian Conference on Computational Geometry*, pages 216–221, 1993.