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

Hee-Kap Ahn† Eunjin Oh‡ Lena Schlipf§ Fabian Stehn¶ Darren Strash‖

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 they 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 s, t where the two persons start.

Keywords: visibility polygon · shortest path · watchman problems

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 [1] and has been extensively studied in computational geometry [2, 3]. Though the problem is NP-hard for polygons with holes [1, 4, 5], 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 of the watchman route problem. Imagine two persons, Romeo and Juliet, travel in a region from their starting locations. They want to minimize the distance they travel in order to see each other. More precisely, given the region and the locations where Romeo and Juliet start, the objective is to compute their paths, one for Romeo and one for Juliet, such that they see each other after traveling along the paths and their travel distances are minimized. This problem can be formally defined 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.*

In the *min-max* variant of the quickest pair-visibility problem, we want to minimize the longer distance that the two points travel to see each other. In the *min-sum* variant, we want to minimize the total travel distance that the two points travel to see each other.

This problem may sound similar to the shortest path problem between s and t, in which the objective is to compute the shortest path $\pi(s, t)$ 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 geodesic distance $|\pi(s, t)|$. We would like to mention that our algorithm to be presented later uses the shortest path as a guide in computing the quickest pair-visibility paths.

*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). An extended abstract of this paper appeared at SWAT'18.

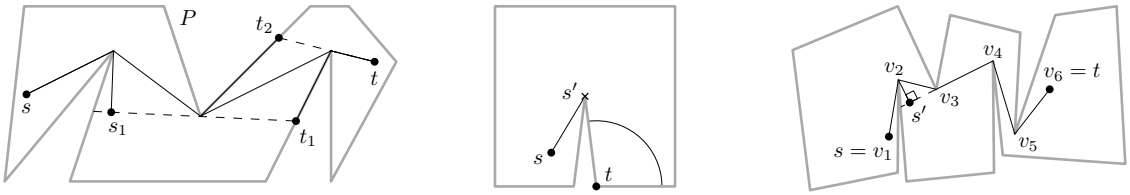
†Department of Computer Science and Engineering, POSTECH, Pohang, South Korea, heekap@postech.ac.kr

‡Max Planck Institute for Informatics, Saarbrücken, Germany, eoh@mpi-inf.mpg.de

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

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

‖Department of Computer Science, Hamilton College, Clinton, New York, USA, dstrash@hamilton.edu



(a) (b) (c)

Figure 1: (a) The quickest pair-visibility problem finds two paths $\pi(s, s_1)$ and $\pi(t, t_1)$ such that $\overline{s_1 t_1} \subset P$ 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:** An instance where $|\pi(s, s')| + |\pi(t, v_4)| = |\pi(s, v_4)| + |\pi(t, v_5)|$. Therefore, both (s', v_4) and (v_4, v_5) are optimal solutions to the min-sum problem.

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 [7]. 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 [8] 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 n is the number of vertices of the polygonal domain, and 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 [9, 10, 11]. 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 supports, for a given query point, finding the shortest path taken from the source point to see the query point efficiently. Khosravi and Ghodsi [10] 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. [9] improved the result and presented an algorithm for the problem in a polygonal domain. Very recently, Wang [11] 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 both min-max and min-sum variants of the quickest pair-visibility problem for a simple polygon. That is, either we want to minimize the maximum length of two traveled paths (min-max) or we want to minimize the sum of the lengths of two traveled paths (min-sum). 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.

2 Preliminaries

Let P be a simple polygon and ∂P be its boundary where $\partial P \subset P$. The vertices of P are given in counterclockwise 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 $\ell \subset P$ 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 the line segment $\overline{pq}$ is contained in P.

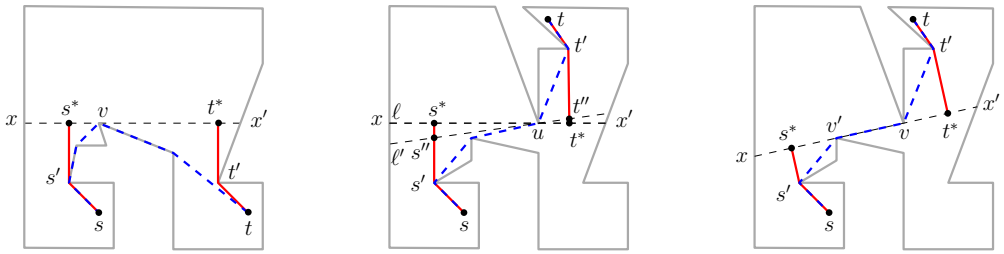


Figure 2: Illustrating cases for the proof of Lemma 1. The examples show the shortest (geodesic) path $\pi(s,t)$ (blue dashed) and the line ℓ tangent to $v \in \pi(s,t)$ that, among all lines tangent to v minimizes shortest paths from s,t to ℓ (red). (a) Both s' and t' lie on the same side of ℓ through s^* and t^*. (b) s' and t' lie on different sides of ℓ. (c) The shortest path $\pi(s,t)$ passes through v and v'.

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).

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 in a closed half-plane bounded by the line containing g. Let $\langle v_0, v_1, \ldots, v_{k-1}, v_k\rangle$ be the sequence of vertices on $\pi(s,t)$ with $s = v_0$ and $t = v_k$.

Lemma 1. *Unless s and t are visible to each other, there is an optimal solution (s^*,t^*) such that $\overline{s^*t^*}$ 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 $\overline{s^*t^*}$. Without loss of generality, assume that $\overline{s^*t^*}$ is horizontal with s^* lying to the left of t^*. Let $\ell = \overline{xx'}$ be the maximal segment contained in P that contains $\overline{s^*t^*}$ with x closer to s^* than to t^*. If $s = s^*$ (or $t = t^*$), then the lemma holds immediately because s (or t) is an endpoint of $\overline{s^*t^*}$. Assume to the contrary that $\overline{s^*t^*}$ contains no vertex of P. Then there are points $p \in P$ in a neighborhood of s^* and $q \in P$ in a neighborhood of t^* such that p and q are visible to each other, and $\max\{|\pi(s,p)|,|\pi(t,q)|\} < \max\{|\pi(s,s^*)|,|\pi(t,t^*)|\}$ and $|\pi(s,p)|+|\pi(t,q)| < |\pi(s,s^*)|+|\pi(t,t^*)|$. This contradicts the optimality of (s^*,t^*).

We now show that $\overline{s^*t^*}$ contains a vertex of $\pi(s,t)$. Let s' be the vertex on $\pi(s,s^*)$ preceding s^* and let t' be the vertex on $\pi(t,t^*)$ preceding t^*. Consider first the case that both s' and t' lie below the line through ℓ. See Figure 2(a). Then ∂P touches $\overline{s^*t^*}$ at a vertex v locally from below. Otherwise, (s^*,t^*) is not optimal by the same argument as in the previous paragraph. Then $s^* \in \overline{xv}$ and $t^* \in \overline{vx'}$. The path $\pi(s,t)$ must cross $\overline{xv}$ at a point y and $\overline{vx'}$ at a point y'. Since y and y' are visible to each other, and $\pi(s,t)$ is a shortest path, $\pi(s,t)$ contains $\overline{yy'}$, which in turn contains v. Thus v lies on $\pi(s,t)$ and $\overline{s^*t^*}$ is tangent to $\pi(s,t)$ at v.

Consider now the case that s' and t' lie on different sides of the line through ℓ. Without loss of generality, assume that s' lies below the line and t' lies above the line. Then $\overline{s^*t^*}$ intersects $\pi(s,t)$. We first show that $\overline{s^*t^*}$ contains an edge of $\pi(s,t)$. Assume to the contrary that $\overline{s^*t^*}$ intersects $\pi(s,t)$ only at a point, say u. Then there is another line segment $\ell' \subset P$ containing u and intersecting both $\overline{s^*s'}$ and $\overline{t^*t'}$. See Figure 2(b). This contradicts that (s^*,t^*) is an optimal solution because, for $s'' = \ell' \cap \overline{s^*s'}$ and $t'' = \ell' \cap \overline{t^*t'}$, $d(s,s'') < d(s,s^*)$ and/or $d(t,t'') < d(t,t^*)$ and s'' and t'' are visible to each other. If u coincides with s^* or t^*, only one of the distance inequalities above holds, we hence consider lexicographic smallest (max,min) solutions in the min-max setting to establish the contradiction. Therefore, $\overline{s^*t^*}$ contains an edge of $\pi(s,t)$, say $\overline{vv'}$. Moreover, one of v and v' touches $\overline{s^*t^*}$ from above, and the other touches $\overline{s^*t^*}$ from below since s' and t' are on different sides of ℓ. See Figure 2(c). Thus, we can assume that ∂P touches $\overline{s^*v}$ at a vertex v' locally from below. Then $\pi(s,t)$ must cross $\overline{xv'}$ at a point y, and $\overline{vx'}$ at a point y'. Since y and y' are visible to each other, and $\pi(s,t)$ is a shortest path, $\pi(s,t)$ contains $\overline{yy'}$, which in turn contains both v' and v. Thus both v' and v lie on $\pi(s,t)$ and $\overline{s^*t^*}$ is tangent to $\pi(s,t)$ at both v' and v. □

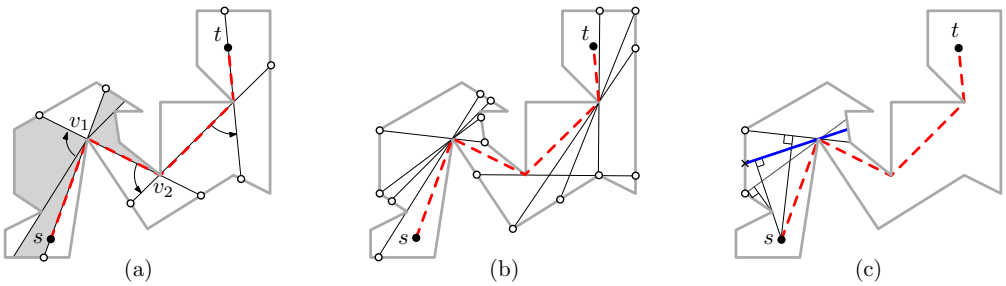


Figure 3: Path-, boundary-, and bend-events. (a) The endpoints of the line-of-sight through $\overline{sv_1}$ make up the first path-event. The line-of-sight rotates until it hits the next path-event: the endpoints of the line-of-sight through $\overline{v_1v_2}$. (b) Boundary-events that are not path-events. (c) A bend-event (marked with a cross) occurs between the two boundary-events. The shortest path from s to the line-of-sight changes at the bend-event.

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

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

Definition 1 (line-of-sight)**.** *We call a segment ℓ a* line-of-sight *if (i) ℓ is a maximal segment contained in P, and (ii) ℓ is tangent to $\pi(s,t)$ at a vertex $v \in \pi(s,t)$.*

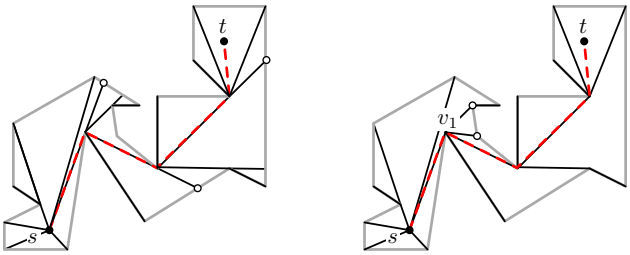
The algorithm we present is in many aspects similar to a sweep-line strategy, except that we do not sweep over the scene in a standard fashion but rotate a *line-of-sight* ℓ in P around the vertices of the shortest path $\pi(s,t)$ in order from s while making use of Lemma 1. Recall that $\langle v_0, v_1, \ldots, v_{k-1}, v_k \rangle$ is the sequence of vertices on $\pi(s,t)$ with $s = v_0$ and $t = v_k$. 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 line-of-sight is rotated around v_i in a way so that it remains tangent to $\pi(s,t)$ at 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 $v_k = t$.

While performing these rotations around the shortest path vertices, we encounter all 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 consider the events as vertices on ∂P unless they are already vertices). Between two consecutive events, we compute the local minima of the relevant distances for the variant at hand in constant time and hence encounter all local 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.



(a) (b)

Figure 4: (a) The shortest path map M_s. All path-events, swept by ℓ^+, appear as endpoints of edges of M_s, except the one induced by v_{k-1} and t; these events are marked with small disks. (b) The shortest path tree T_s. The boundary-events, swept by ℓ^+, tangent to $\pi(s,t)$ at v_1 are marked. Clearly, the parent vertex of these vertices in T_s is v_1.

Lemma 2 (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 of their occurrence 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 [12]. 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 (see also Figure 4(a)). For each index i with $0 < i < k$, 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. Note that the path-event induced by v_{k-1} and t is an exception, but it can also be computed in $O(1)$ time during the process by considering the triangle of M_s that contains t.

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 5.) Observe that

P1 all A_i^+ for $0 < i < k$ are pairwise disjoint in their interior,
P2 all A_i^- for $0 < i < k$ are pairwise disjoint in their interior,
P3 for all $0 < i < k$ and all points $p \in A_i^+$ the shortest path $\pi(s,p)$ contains v_i (i.e., v_i is the predecessor of p on $\pi(s,p)$),
P4 for all $0 < i < k$ and all points $p \in A_i^-$ the shortest path $\pi(p,t)$ contains v_i (i.e., v_i is the successor of p on $\pi(p,t)$).

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 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

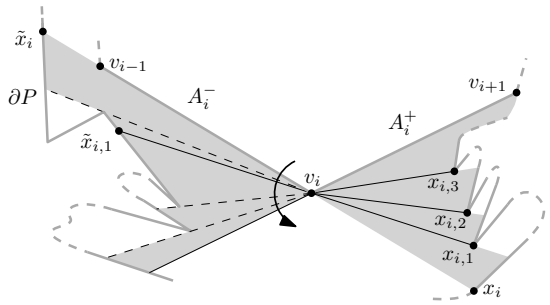


Figure 5: Let $E_i^+ = \langle x_{i,1}, \ldots, x_{i,j} \rangle$ for an index $1 \leq j \leq n$. We start at $\tilde{x}_i$ and follow the (triangular) cells of M_t incident to v_i in counter-clockwise order around v_i until we find $\tilde{x}_{i,1}$. Then we continue to follow such cells until we find $\tilde{x}_{i,2}$, and so on.

∂P and checking the vertices within the candidate set (see Figure 4(b)). We compute the boundary-events that are vertices of P swept by ℓ^- in a similar way.

So far we have 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 5. 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 of the shortest path map M_t of 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 3. *All bend-events can be computed in $O(n)$ time, sorted in the order as they appear on the boundary of P.*

Proof. Bend-events occur between consecutive path- and boundary-events; they can also coincide with these events. 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 lines-of-sight) 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. The path- and boundary-events define the area which is swept over by ℓ. Thus, we know which positions for ℓ we have to consider in order to compute all bend-events.

As in the proof of Lemma 2, 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 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 $\pi(s,\ell) = \pi(s,\ell^-)$ over the events.

While ℓ rotates around v, the combinatorial structure of $\pi(s,\ell)$ may change. Specifically, let $e_\ell = \overline{uw}$ 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 e_ℓ hits a vertex u' of P (if u' at this event is an endpoint of e_ℓ, then this bend-event coincides with a previously computed boundary-event) and splits into two edges sharing u' (an event of type **T1**) or the two edges of $\pi(s,\ell)$ incident to u become parallel (an event of type **T2**). (Then they merge into

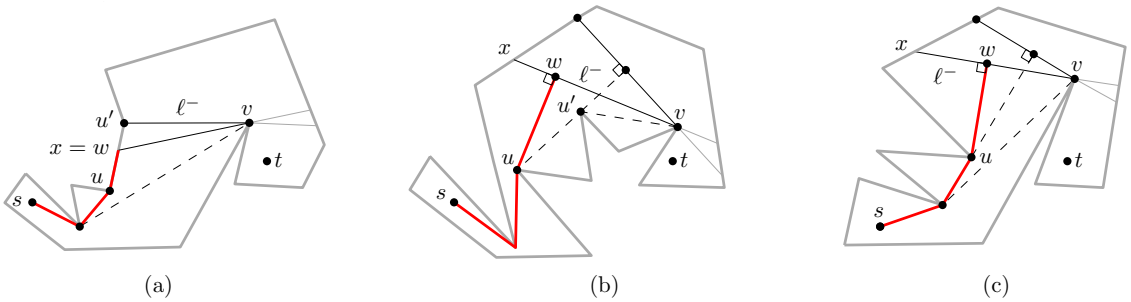


Figure 6: (a) A bend-event of type **T1** occurs when $x = u_\ell$ reaches u'. (b) A bend-event of type **T1** occurs when $e_\ell = \overline{uw}$ hits a vertex u' of $\pi(s,t)$. (c) A bend-event of type **T2** occurs when two edges incident to u are parallel.

one and u disappears from $\pi(s,\ell)$.) See Figure 6. 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 a vertex on ∂P as it represents a bend-event.

Lemma 4. *An event of type **T1** occurs only when (i) x reaches a vertex u', or (ii) e_ℓ hits a vertex u' of $\pi(s,t)$ in its interior. Moreover, for case (ii), u and u' are consecutive in $\pi(s,t)$.*

Proof. Imagine ℓ is rotated around v infinitesimally further from the current event. Then either e_ℓ is orthogonal to ℓ or not. If e_ℓ is not orthogonal to ℓ, 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 6(a).

Now consider the case that e_ℓ is orthogonal to ℓ. Notice that the shortest path from a vertex v to a segment within a simple polygon lies inside a *funnel*, a region bounded by the shortest paths from v to both endpoints of the segment and the segment. For more details see [12]. Thus, u' is contained in $\pi(u,v)$. See Figure 6(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 5. *Once a vertex disappears from $\pi(s,\ell)$, it never appears again on the shortest path during the rotation of the 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 distinct lines-of-sight ℓ_1 and ℓ_2 during the rotation. First note that if u is an endpoint of $\pi(s,\ell_1)$ (or $\pi(s,\ell_2)$), it is a boundary- and bend-event, and would only appear once when rotating the line-of-sight. Therefore, both $\pi(s,\ell_1)$ and $\pi(s,\ell_2)$ must contain u in their interiors, and both of them also contain $\pi(s,u)$ in their interiors. Since u disappears from $\pi(s,\ell_1)$, the edge of $\pi(s,\ell_1)$ incident to u (on $\pi(u,\ell_1)$) is orthogonal to ℓ_1. We claim that u appears on $\pi(s,\ell_2)$ due to case (ii) of type **T1**, 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 (i) of type **T1**. 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 $\pi(s,\ell_1)$. By Property **P2**, ℓ^- sweeps a vertex only once. Thus, u appears on $\pi(s,\ell_2)$ due to case (ii) of type **T1**, 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 u_1 and u_2, respectively. Without loss of generality, assume that u_1 is closer to s than u_2. We show that $\pi(p_1,p_2)$ contains u_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. Since both u_1 and u_2 are vertices of $\pi(s,t)$, $\pi(s,u_2)$ contains u_1. Let p be the point on ℓ_2^- farthest from u_2 such that $\pi(s,p)$ contains u_1. Since the boundary of P intersect neither $\overline{u_1p}$ nor $\overline{u_2p}$, $\pi(u_1,u_2)$ is contained in the triangle with corners u_1, u_2, p. No line segment parallel to ℓ_2 is tangent to $\pi(s,t)$ at u_1, which is a contradiction. Therefore, $\pi(s,p_2)$ contains u_1 for any point $p_2 \in \ell_2$. Then since ℓ_1 is tangent to $\pi(s,t)$, $\pi(p_1,p_2)$ contains u_1 for any two points $p_1 \in P_1$ and $p_2 \in \ell_2$. Thus, $\pi(s,\ell_2)$ contains $\pi(s,u_1)$, and no vertex in P_1 other than the vertices of $\pi(s,u_1)$ 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. □

Using the two lemmas, we can compute all bend-events as follows. For a line-of-sight ℓ rotating around a vertex v, we have three candidates for the next bend-event. Let e be the edge of P containing the endpoint of ℓ^- other than v, and let u' be the neighboring vertex of u in $\pi(u,t)$. The next bend-event is (1) the endpoint of e not contained in $\pi(s,\ell)$ if it exists, (2) the intersection point between e and the line through v and orthogonal to uu' if it exists, or (3) the intersection point between e and the line through v and orthogonal to u'' if it exists, where u'' is the neighboring vertex of u in $\pi(s,\ell)$ closer to s. Note that the first two cases are type **T1** events and the last case is a type **T2** event. We can compute all of the three events in constant time. Also, we can update u, e_ℓ, x and $\pi(s,\ell)$ accordingly in constant time. Therefore, the time for computing all bend-events is linear in the amount of the combinatorial change on $\pi(s,\ell)$. By Lemma 5, the the amount of the combinatorial change is $O(n)$, and therefore, we can compute all bend-events in $O(n)$ 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 6. *For any two consecutive events, the line-of-sight ℓ lying between them that minimizes the sum or the maximum 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. We assume that $\mathcal{L}$ contains no vertical line-of-sight. Otherwise, we consider the subset containing all lines-of-sight with positive slopes, and then the subset containing all lines-of-sight with negative slopes.

These lines-of-sight share a common vertex v of $\pi(s,t)$. We will give an algebraic function for $|\pi(s,\ell)|$ for $\ell \in \mathcal{L}$. An algebraic function for $|\pi(t,\ell)|$ can be obtained analogously. Observe that $\pi(s,u)$ is the same for all $\ell \in \mathcal{L}$, where u is the second to the last vertex u of $\pi(s,\ell)$ from s. Thus, we consider only the length of $\pi(u,\ell)$, which is a line segment. 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.

Let $\ell(\alpha)$ denote the line of slope α passing through v for $\alpha > 0$, which is represented as $y = \alpha x + f(\alpha)$, where $f(\alpha)$ is a function linear in α. Then the distance between u and $\ell(\alpha)$ can be represented as $|c_1\alpha + c_2|/\sqrt{\alpha^2+1}$, where c_1 and c_2 are constants depending only on v and u. Thus, our problem reduces to finding a minimum of the function of the form $(|c_1\alpha + c_2| + |c_1'\alpha + c_2'|)/\sqrt{\alpha^2+1}$ and $\max(|c_1\alpha + c_2|, |c_1'\alpha + c_2'|)/\sqrt{\alpha^2+1}$, respectively, for four constants c_1, c_2, c_1' and c_2', and for all α such that $\ell(\alpha)$ contains a line-of-sight in $\mathcal{L}$. We can find a minimum in constant time using elementary analysis. □

Theorem 1. *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 2. 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 3, 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 6. It maintains the smallest one among the local optima computed so far, and returns 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^*) (Lemma 1). 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. □

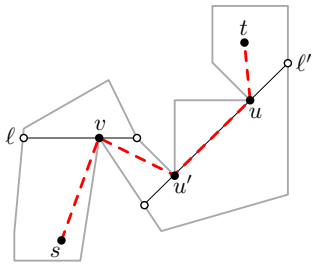


Figure 7: Let ℓ be a be a line-of-sight which is tangent to $\pi(s,t)$ at a vertex v. And let ℓ' be be a line-of-sight that comes after ℓ during the rotational sweep process. Clearly, $|\pi(s,\ell')| \geq |\pi(s,\ell)|$.

Corollary 1. *By the same algorithm, one can also compute optimal pairs (s^*, t^*) that minimize*

- $\max(\lambda|\pi(s,s^*)|, (1-\lambda)|\pi(t,t^*)|)$ *for some* $0 \leq \lambda \leq 1$,
- $\max(\alpha + |\pi(s,s^*)|, \beta + |\pi(t,t^*)|)$ *for some* $\alpha, \beta \in \mathbb{R}^+$.

The first modification introduced in Corollary 1 models that Romeo and Juliet travel with different speeds. It is easy to see, that this formulation is equivalent to minimizing the objective $\max(\alpha|\pi(s,s^*)|, \beta|\pi(t,t^*)|)$ for some $\alpha, \beta \in \mathbb{R}^+$. The second variant can be motivated as follows: Imagine Romeo (and Juliet) is driving a car that before departing from s (and t) already drove a distance of α (and β). The objective $\max(\alpha + |\pi(s,s^*)|, \beta + |\pi(t,t^*)|)$ minimizes the largest distance any of the two cars had to drive in order to establish a line-of-sight.

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 [13, 14]. 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 [14]. Thus, we can apply a 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 a 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 7. *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 that 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 2. 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 $\pi(s,t)$. See Figure 7.

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 $\langle v_0, v_1, \ldots, v_{k-1}, v_k \rangle$ be the sequence of vertices on $\pi(s,t)$ with $s = v_0$ and $t = v_k$. 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. We use the data structure for ray shooting given by Hershberger and Suri [15] with linear preprocessing and logarithmic query time. Then, we compute the length of $\pi(s,\ell)$ and $\pi(t,\ell)$ in $O(\log n)$ time using the data structure given by Guibas and Hershberger for computing the distance between a query point and a query line segment in $O(\log n)$ time [13]. 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 5. 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 [15]. 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 between $\tilde{x}_i$ and $\tilde{x}_{i+1}$ in counter-clockwise order, and γ be the part of ∂P lying between x_i and 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 a 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 8(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 [15]. Then we compare $|\pi(s,\ell)|$ and $|\pi(t,\ell)|$ in $O(\log n)$ time. The point x^* comes after u from x_i if and only if $|\pi(s,\ell)| < |\pi(t,\ell)|$. 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 a 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 [14]. See Figure 8(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. Notice that π' is a path from ℓ_1 to ℓ_2, concatenating the two shortest paths from ℓ_1 to s' and from s' to ℓ_2.

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 (because

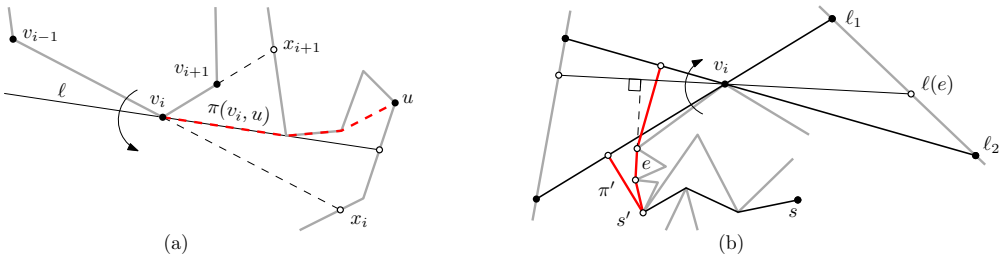


Figure 8: (a) The line-of-sight intersecting $\pi(v_i, u)$ contains the edge of $\pi(v_i, u)$ incident to v_i. (b) The maximal common subpath of $\pi(s, \ell_1)$ and $\pi(s, \ell_2)$ from s is $\pi(s, s')$; $\pi' = \pi(s', \ell_1) \cup \pi(s', \ell_2)$ (blue).

the order of the edges of π', as they appear on the path, are radially sorted around v_i). 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 a 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 contain ℓ^* between them. Recall that the combinatorial structure of $\pi(s, \ell)$ (and $\pi(t, \ell)$) is the same for every line-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 $u_s, u_t, |\pi(s, u_s)|$ and $|\pi(t, u_t)|$ in $O(\log n)$ time. Then we apply the algorithm in Lemma 6 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 2. *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 $s, t \in P$.*

6 Conclusions and Open Problems

We have presented a linear time algorithm that solves two variants of the quickest pair-visibility problem for a simple polygon: either we want to minimize the maximum length of a traveled path or we want to minimize the sum of the lengths of both traveled paths.

Additionally, we have considered a query version of the quickest-visibility 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.

We conclude this paper with some interesting open problems.

1. Is there a way to extend our algorithm to more than two query points? More precisely, given k points in a simple polygon, compute the minimum distance that these points must travel in order to see each other (at the same moment).

2. Find an efficient algorithm for the query version of the quickest-visibility problem for the min-sum problem.

Acknowledgments

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

References

[1] W. Chin, S. Ntafos, Optimum watchman routes, in: Proceedings of the 2nd ACM Symposium on Computational Geometry, SoCG 1986, 1986, pp. 24–33. doi:10.1145/10515.10518.

[2] S. Carlsson, H. Jonsson, B. J. Nilsson, Finding the shortest watchman route in a simple polygon, Discrete & Computational Geometry 22 (3) (1999) 377–402. doi:10.1007/PL00009467.

[3] J. S. B. Mitchell, Approximating watchman routes, in: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, 2013, pp. 844–855. doi:10.1137/1.9781611973105.60.

[4] W. Chin, S. Ntafos, Optimum watchman routes, Information Processing Letters 28 (1) (1988) 39–44. doi:10.1016/0020-0190(88)90141-X.

[5] A. Dumitrescu, C. D. Tóth, Watchman tours for polygons with holes, Computational Geometry 45 (7) (2012) 326–333. doi:10.1016/j.comgeo.2012.02.001.

[6] M. Dror, A. Efrat, A. Lubiw, J. S. B. Mitchell, Touring a sequence of polygons, in: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC '03, 2003, pp. 473–482. doi:10.1145/780542.780612.

[7] A. Ganguli, J. Cortes, F. Bullo, Visibility-based multi-agent deployment in orthogonal environments, in: Proceedings of the 2007 American Control Conference (ACC '07), 2007, pp. 3426–3431. doi:10.1109/ACC.2007.4283034.

[8] E. L. Wynters, J. S. B. Mitchell, Shortest paths for a two-robot rendez-vous, in: Proceedings of the 5th Canadian Conference on Computational Geometry, 1993, pp. 216–221.

[9] E. M. Arkin, A. Efrat, C. Knauer, J. S. B. Mitchell, V. Polishchuk, G. Rote, L. Schlipf, T. Talvitie, Shortest path to a segment and quickest visibility queries, Journal of Computational Geometry 7 (2) (2016) 77–100. doi:10.20382/jocg.v7i2a5.

[10] R. Khosravi, M. Ghodsi, The fastest way to view a query point in simple polygons, in: Proceedings of the 21st European Workshop on Computational Geometry, 2005, pp. 187–190.

[11] H. Wang, Quickest visibility queries in polygonal domains, in: Proceedings of the 33rd International Symposium on Computational Geometry (SoCG 2017), Vol. 77 of Leibniz International Proceedings in Informatics (LIPIcs), 2017, pp. 61:1–61:16. doi:10.4230/LIPIcs.SoCG.2017.61.

[12] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, R. E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, Algorithmica 2 (1–4) (1987) 209–233. doi:10.1007/BF01840360.

[13] L. J. Guibas, J. Hershberger, Optimal shortest path queries in a simple polygon, Journal of Computer and System Sciences 39 (2) (1989) 126–152. doi:10.1016/0022-0000(89)90041-X.

[14] J. Hershberger, A new data structure for shortest path queries in a simple polygon, Information Processing Letters 38 (5) (1991) 231–235. doi:10.1016/0020-0190(91)90064-O.

[15] J. Hershberger, S. Suri, A pedestrian approach to ray shooting: Shoot a ray, take a walk, Journal of Algorithms 18 (3) (1995) 403–431. doi:10.1006/jagm.1995.1017.