Trajectory Visibility at First Sight

Mohammad Ali Abam
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Mohammad Ghodsi
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Seyed Mohammad Hussein Kazemi[1]
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
LaBRI, University of Bordeaux, Bordeaux, France

Abstract

Let P be a simple polygon with n vertices, and let two moving entities $q(t)$ and $r(t)$ travel at constant (possibly distinct) speeds v_q and v_r along line-segment trajectories τ_q and τ_r inside P. We study the exact first-visibility time $t^* = \min\left\{ t \geq 0 \mid \overline{q(t)\, r(t)} \subseteq P \right\}$, the earliest moment at which the segment joining $q(t)$ and $r(t)$ lies entirely within P.

Prior work by Eades et al. [9, 10] focused on this question in the setting of a simple polygon. They gave a one-shot decision algorithm running in $\mathcal{O}(n)$ time. For a stationary entity and a moving one, they suggested a structure that, after $\mathcal{O}(n \log n)$ pre-processing, answers the decision query in $\mathcal{O}(\log n)$ time, requiring $\mathcal{O}(n)$ space. In addition, for moving entities, after preprocessing time of $\mathcal{O}(n \log^5 n)$, they construct a data structure with $\mathcal{O}(n^{3/4} \log^3 n)$ query time and $\mathcal{O}(n \log^5 n)$ space. Variants for polygonal domains with holes or when entities cross the boundary of P lie beyond our scope.

In this work, we go beyond the decision to compute t^* exactly under three models for a simple polygon P. When both trajectories are known in advance, we preprocess P in $\mathcal{O}(n)$ time and space and thereafter answer each query in $\mathcal{O}(\log n)$ time. If one trajectory τ_r is fixed while τ_q is given as query, we build a structure in $\mathcal{O}(n \log n)$ time and space that computes t^* in $\mathcal{O}(\log^2 n)$ time per query. In a setting where the trajectories are not known in advance, we develop a randomized structure with $\mathcal{O}(n^{1+\varepsilon})$ expected pre-processing time and $\mathcal{O}(n)$ space, achieving an $\mathcal{O}(n^{1/2}\, \mathrm{polylog}(n))$ expected query time for any fixed $\varepsilon > 0$.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases trajectories, visibility, semi-algebraic range searching

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

The analysis of moving entities constitutes a broad research area, ranging from robotics and geographic information science (GIScience) to meteorology and ecology [22, 4, 15, 19, 8]. A body of work focuses on extracting features from historical trajectory data, such as identifying when entities have passed closely together or detecting groups that move in formation [13]. However, a class of problems emerges when the goal is not to analyse the past, but to predict future events for entities on known or planned paths. This is especially true in environments with obstacles or defined boundaries, which mirrors the constrained polygonal setting of our work.

Practical applications of such predictive line-of-sight analysis are numerous. In robotics and automated systems, for example, determining when two microrobots can establish direct communication [21] or when an autonomous vehicle will gain a clear view of a target is a fundamental task. Similar challenges arise in biology when tracking potential interactions

[1] corresponding author

within animal colonies in complex habitats [3] and in optimising sensor-based systems, such as dual-axis solar trackers that must maintain an unobstructed path to the sun [16]. As the complexity and autonomy of these systems grow, the demand for a robust theory of trajectory visibility and the development of efficient, provable algorithms naturally escalates.

1.1 Preliminaries

Let P be a simple polygon with n vertices, and let ∂P denote the boundary of P, which consists of its vertices and edges. A point $p \in P$ is considered to be *weakly visible* from a segment $s \subset P$ if there exists a point $q \in s$ such that the segment $\overline{pq}$ is entirely contained within P. Similarly, two segments s_1 and $s_2 \subset P$ are weakly visible to each other if, for every point in one segment, there exists a corresponding point in the other segment such that the connecting segment is also contained within P [5, 12]. For a given query point $p \in P$, the *visibility polygon* of p, denoted by $V(p)$, is defined as

$$V(p) = \{q \in P \mid \overline{pq} \subseteq P\}.$$

It is well known that $V(p)$ can be computed in $O(n)$ time using a rotational sweep algorithm [18]. Given a segment $s \subset P$, the *weak visibility polygon* of s is defined as

$$W(s) = \{q \in P \mid \exists\, r \in s \text{ such that } \overline{qr} \subseteq P\}.$$

Using similar techniques to those used for the visibility polygon, $W(s)$ can also be computed in $O(n)$ time [12]. Additionally, a *ray shooting* query is defined as follows: given a query ray r with origin $p \in P$, the goal is to determine the first intersection point between r and the boundary ∂P. After a linear-time preprocessing phase, such queries can be answered in $\mathcal{O}(\log n)$ time [6].

Splinegons (also known as curved polygons) are considered extensions of traditional polygons. A splinegon $\mathcal{S}$ is created from a polygon P by substituting one or more of its edges with curved edges, ensuring that the area enclosed by each curved edge and the line segment connecting its endpoints remains convex [7]. Provided a simple splinegon S with n edges, there exists a data structure requiring $\mathcal{O}(n)$ preprocessing time and $\mathcal{O}(n)$ space such that for any query point p and a ray $\overrightarrow{r}$ emanating from p, the first intersection of $\overrightarrow{r}$ with $\mathcal{S}$ can be reported in $\mathcal{O}(\log n)$ time [20, 11].

Consider a point $p \in P$. For every vertex v of the polygon P, let $\pi(p, v)$ denote the shortest Euclidean path inside P connecting p and v. The shortest-path tree rooted at p, denoted $T(p)$, is the union of these paths:

$$T(p) = \bigcup_{v\ \in P} \pi(p, v).$$

Equivalently, $T(p)$ is the tree formed by taking, for every polygon vertex v, the unique polygonal path of minimum length from p to v, where all intermediate vertices on each $\pi(p, v)$ are reflex vertices of P [2]. Let us assume all paths in $T(p)$ are stored as *two-way linked lists*.

For two line segments $s_1, s_2 \subset P$ let $L(s_1, s_2)$ be their visibility glass. The $L(s_1, s_2)$ comprises the (potentially empty) collection of all line segments that lie between s_1 and s_2 within P. When $L(s_1, s_2)$ is not empty, there exist segments $s_1^{'} \subset s_1$ and $s_2^{'} \subset s_2$ such that $L(s_1, s_2)$ corresponds to the hourglass [14] formed by $s_1^{'}$ and $s_2^{'}$. The segments $s_1^{'}$ and $s_2^{'}$ are constrained by two bi-tangents along the shortest paths connecting the endpoints of s_1 and s_2. The total running time of constructing $L(s_1, s_2)$ is proportional to $\mathcal{O}(n)$ [9]. For every pair of segments s_1 and s_2, denote by $p^+_{s_1}$, $p^-_{s_1}$ and $p^+_{s_2}$, $p^-_{s_2}$ the upper and lower endpoints of

Table 1 Preprocessing, storage, and query-time complexities for the trajectory visibility problem in a simple polygon with line-segment trajectories. The first two rows list the best-known data-structure bounds from P. Eades et al. [9] for deciding whether two moving entities ever become visible (referred to as **Decision**). The last three rows summarise our new results (Theorems 1, Theorems 2, Theorems 3), which determine the earliest moment at which mutual visibility occurs (referred to as **Finding** t^*).

Reference	Scenario	Preprocessing	Space	Query time
[9] (Sec. 5)	Stationary & moving (**Decision**)	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
[9] (Sec. 4)	Query trajectories (**Decision**)	$\mathcal{O}(n \log^5 n)$	$\mathcal{O}(n \log^5 n)$	$\mathcal{O}\big(n^{3/4} \log^3 n\big)$
Theorem 1	Fixed trajectories (**Finding** t^*)	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
Theorem 2	One trajectory as query (**Finding** t^*)	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log^2 n)$
Theorem 3	Query trajectories (**Finding** t^*)	$\mathcal{O}\big(n^{1+\varepsilon}\big)$ (expected)	$\mathcal{O}(n)$	$\mathcal{O}\big(n^{1/2} \log^B n\big)$ (expected)

s_1 and s_2, respectively. Let $\gamma^+ = \pi(p^+_{s_1}, p^+_{s_2})$ be the *upper chain* and $\gamma^- = \pi(p^-_{s_1}, p^-_{s_2})$ the *lower chain* of the hourglass of s_1 and s_2.

In their work on semi-algebraic range searching, Agarwal et al. [1] fixed constants d, Δ, s and an arbitrary $\varepsilon > 0$, and showed that an arbitrary n-point set in $\mathbb{R}^d$ admits a data structure with *expected* preprocessing time $\mathcal{O}(n^{1+\varepsilon})$, and the space of $\mathcal{O}(n)$, that answers any constant–complexity semi-algebraic-range query in $\mathcal{O}\big(n^{1-1/d} \log^B n\big)$ s.t. $B = B(d, \Delta, s, \varepsilon)$ [2].

Let a trajectory be a sequence of time-stamped locations in $\mathbb{R}^d$, which models the movement of an entity in a polygon. The problem of trajectory visibility was explored by P. Eades et al. [9]. Informally, given a simple polygon or a polygonal domain $P^{'}$, and the paths of two moving entities q and r, determine if there is ever a time at which q and r can see each other. Certainly, there are various scenarios based on whether $P^{'}$ is a simple polygon or a polygonal area, and whether the trajectories cross $\partial P^{'}$ or not. Throughout this paper, we assume that the entities are moving in line segments trajectories at possibly different constant speeds and cannot see through the edges of $P^{'}$. For the sake of notation, suppose that two entities q and r move on trajectories $\tau_q, \tau_r \subset P^{'}$. Eades et al. emphasized identifying whether there is ever a time when the two entities can see each other. That enables a temporal breakdown of the problem: a conclusive answer of *no* is determined if it is *no* for all pairs of successive timestamps [9].

1.2 Our Contributions

From now on, we only consider a simple polygon P, and that $\tau_q, \tau_r \subset P$, that is, they never cross ∂P. Specifically, our work aims to solve some of the directions left for future explorations in [9] (see also [10]). We aim to determine the first time t^* at which q and r can see each other. We emphasize that in the *most general case* explored in this paper, the constant speeds of q and r, denoted as v_q and v_r, together with the trajectories of q and r, are not given alongside P, but provided as *queriers* (see also Table 1 in [9]). More specifically, we address the following (the readers may refer to Table 1 as well):

1. Suppose that two moving entities q and r with fixed line-segment trajectories are given. For query speeds v_q and v_r, determine the first time t^* when they become mutually

[2] This can be $\mathcal{O}(n^{1-1/d+\varepsilon})$ if we assume D_0-general position for every point.

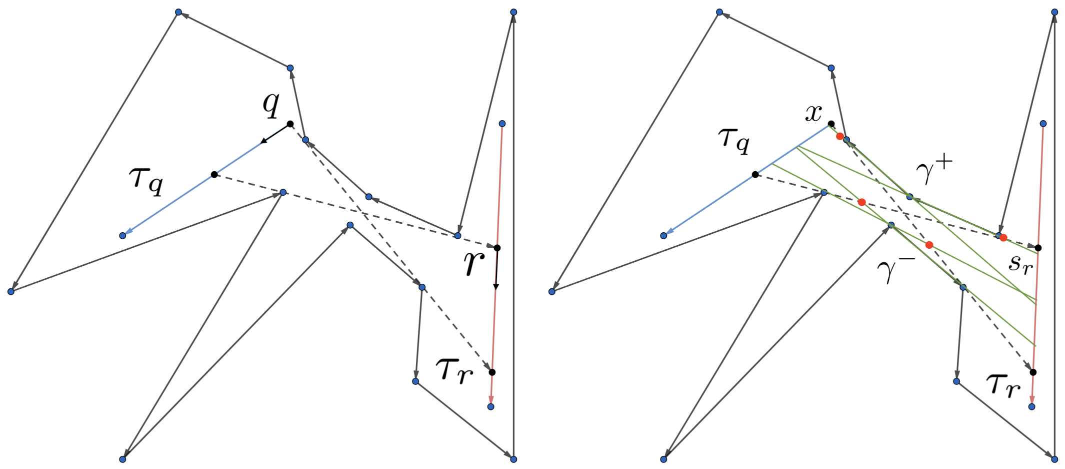


Figure 1 (Left) Given two moving entities q and r with their trajectories τ_r and τ_q in red and blue within a simple polygon, the dotted lines indicate the bi-tangents of their visibility glass. (Right) Extensions of segments between consecutive reflex vertices (in green) (known as critical constraints [2]) intersect their bi-tangents (red dots). For every $x \in I_q$, there is a corresponding subsegment $s_r \in \tau_r$ such that every point in s_r is visible to x within P.

visible. We achieve a query time of $\mathcal{O}(\log n)$ after $\mathcal{O}(n)$ preprocessing. Moreover, our approach can naturally extend to reporting the entire set of time intervals during which the entities are mutually visible.

2. We extend the above case by assuming that q's trajectory is not known in advance and is provided only as a query, along with v_q and v_r. We develop a data structure that, after $\mathcal{O}(n \log n)$ preprocessing, answers queries in $\mathcal{O}(\log^2 n)$ time, suing $\mathcal{O}(n)$ space.
3. We finally remove all the above relaxations, meaning none of q's and r's trajectories are known in advance, same as v_q and v_r. We develop a data structure that, after $\mathcal{O}(n^{1+\varepsilon})$ *expected* pre-processing time, answers queries in $\mathcal{O}(n^{1/2} \log^B n)$ *expected* time, where B is a constant. Our data structure requires $\mathcal{O}(n)$ space.

We note a recent similar attempt [17], but our approach does not overlap with theirs. While they have explored a similar problem, our investigations suggest that our approach resolve serious issues in their analysis.

2 Fixed Trajectories

Consider the $L(\tau_q, \tau_r)$. For every pair of consecutive reflex vertices v_i and v_{i+1} on its upper chain γ^+ (and similarly on γ^-), let $l(v_i, v_{i+1})$ be the line through them. Let the ray emitting from v_i to v_{i+1} on $l(v_i, v_{i+1})$ refer to the ray with origin at v_i directed along $l(v_i, v_{i+1})$ toward v_{i+1} (similarly for the ray from v_{i+1} to v_i). Let each of these rays be followed until its first intersection with ∂P (excluding v_i and v_{i+1}). The closed segment of $l(v_i, v_{i+1})$ between those two intersection points on ∂P is denoted as a *critical constraint* by Aronov et al. [2] and others before them. Define:

$$I_q = \bigcup_i \left(l(v_i, v_{i+1}) \cap \tau_q\right) \quad \text{and} \quad I_r = \bigcup_i \left(l(v_i, v_{i+1}) \cap \tau_r\right)$$

These intersection points partition τ_q and τ_r into subsegments. For each intersection point $x \in I_q$, there is a corresponding subsegment $s_r \subset \tau_r$ such that every point in s_r is visible to

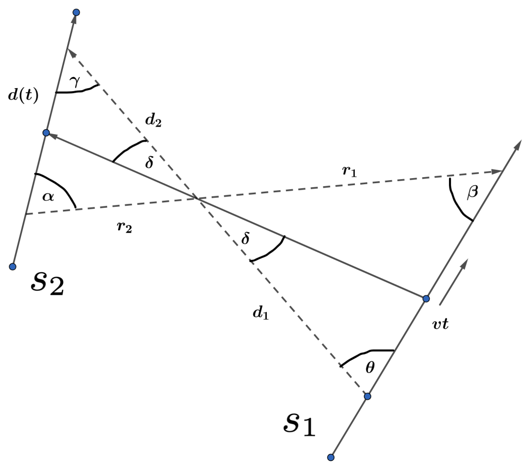


Figure 2 An entity travels along segment s_1 at constant speed v, covering distance vt from its start. The ray from this entity to segment s_2 meets the hourglass bi-tangent at angle δ, while the fixed angles between the bi-tangent and the upper and lower chains are θ and γ, respectively. Distances d_1 and d_2 are measured from the endpoints of s_1 and s_2 to the corresponding tangency points.

x within P (see Figure 1).

Next, we define a mapping to map positions along τ_q and τ_r into a two-dimensional diagram $D \subset \mathbb{R}^2$. Let $\varphi_q : \tau_q \to [0, \operatorname{len}(\tau_q)]$ and $\varphi_r : \tau_r \to [0, \operatorname{len}(\tau_r)]$ be the arc-length parametrizations of τ_q and τ_r, respectively. Note that we map the starting point of the trajectories to 0. In D, the x-coordinate is determined by φ_q and the y-coordinate by φ_r. Each $x \in I_q$ corresponds to a vertical segment in D that represents a segment along τ_r on which every point is visible to x (see Figure 3).

Observe that the mapping from positions on τ_q to positions on τ_r is non-linear. For example, an entity moving along a segment s_1 at constant speed v maps to a point on another segment s_2 via a nonlinear function $f(t)$, where the distance $d(t)$ from a fixed endpoint of s_2 is strictly convex.

▶ **Corollary** (1). *Let $s_1, s_2 \subset P$ be two line segments. Suppose an entity moves with constant speed v on s_1. Construct $L(s_1, s_2)$ and consider its intersection points. Define a mapping of the entity's position as $f(t)$ to be a point on s_2 at distance $d(t)$ from a certain endpoint of s_2 at time t. Then, as the entity moves, the mapping $f(t)$ transitions between the intersection points, and $d(t)$ is strictly convex and non-linear in t.*

Proof. Let $d(t)$ be denoted as x. Referring to Figure 2 and applying the sine law on the two relevant triangles, we obtain: $v\,t = \dfrac{d_1 \sin\delta}{\sin(\delta+\theta)}$ and $x = \dfrac{d_2 \sin\delta}{\sin(\gamma+\delta)}$.

Eliminating δ gives the closed-form: $x = \dfrac{d_2\, v\, t\, \sin\theta}{d_1 \sin\gamma \;+\; v\, t\, \sin(\theta-\gamma)}$.

Now set $A = \dfrac{v\ \sin(\theta-\gamma)}{d_1}$, $B = \dfrac{d_1 \sin\gamma}{v\ \sin\theta}$, so that $x = d_2 \cdot \dfrac{t}{B} - d_2 \cdot \dfrac{At^2}{B(At+B)}$

A straightforward differentiation shows that the second derivative of x with respect to t is strictly positive (or negative) under non-degeneracy conditions (e.g., $d_1 > vt\cos\theta$ and non-zero angles), which establishes that $d(t)$ is strictly convex. ◀

Given query speeds v_q and v_r for q and r, their positions become $\varphi_q(t) = v_q\, t$ and

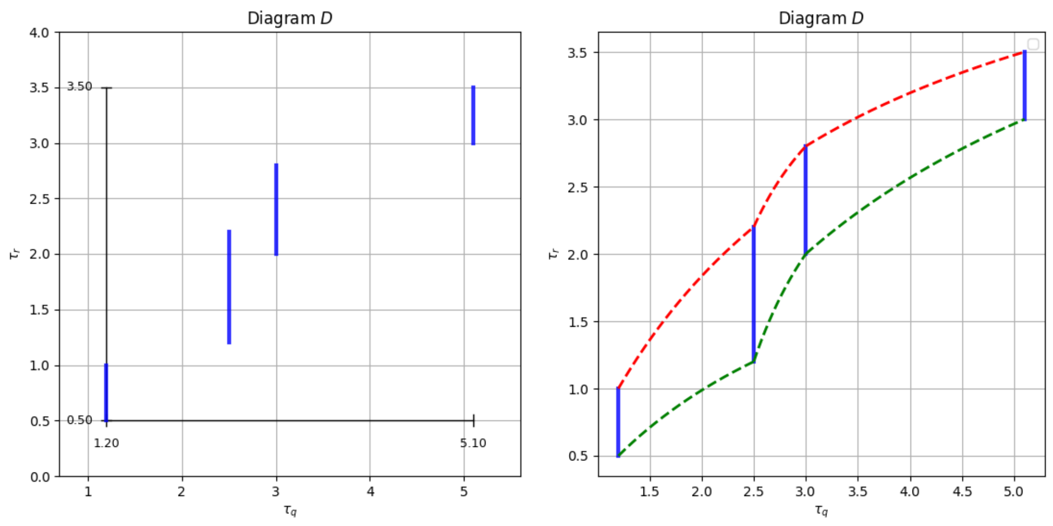


Figure 3 (Left) Vertical segments in the diagram D induced by the intersection points I_q on τ_q: each $x = \varphi_q(x_i)$ spans the interval of φ_r-values for which points on τ_r are visible to x_i. (Right) The curved boundary (splinegon) obtained by the nonlinear mapping discussed in (Corollary 1 at) Section 2

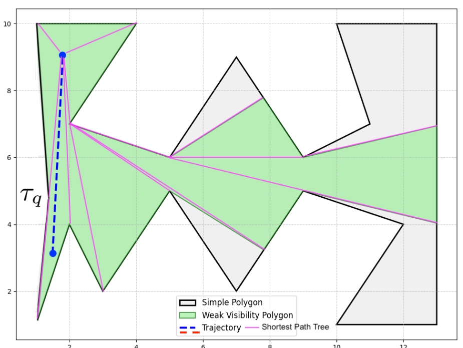
$\varphi_r(t) = v_r\,t$. In D, consider the ray $\overrightarrow{r}$ from the origin with slope $\tan\alpha = \frac{v_q}{v_r}$. The first time t^* of mutual visibility occurs when the point $(v_q\,t, v_r\,t)$ lies on (or above) the curve boundary (according to Corollary 1) of D. By Corollary 1, the image in D of any constant-speed traversal of a straight subsegment of τ_q or τ_r is an algebraic arc (coordinates given by rational functions of t of constant degree). Together with the vertical segments arising from the mapped critical-constraint intersections, these arcs form the boundary of the mutual-visibility region in D. Each arc endpoint coincides with a mapped critical-constraint intersection, and there are only $\mathcal{O}(n)$ such intersections overall; hence the boundary is a closed chain of $O(n)$ piecewise-algebraic arcs. This boundary is therefore a *splinegon*. Ray shooting on the splinegon in D after a preprocessing in $\mathcal{O}(n)$ time yields t^* in $\mathcal{O}(\log n)$ time. Therefore, the following theorem holds:

▶ **Theorem 1.** *Constructing D in $\mathcal{O}(n)$ time, using $\mathcal{O}(n)$ space, one can process any query (v_q, v_r) to determine in $\mathcal{O}(\log n)$ time the smallest $t^* \geq 0$ for which q and r become mutually visible.*

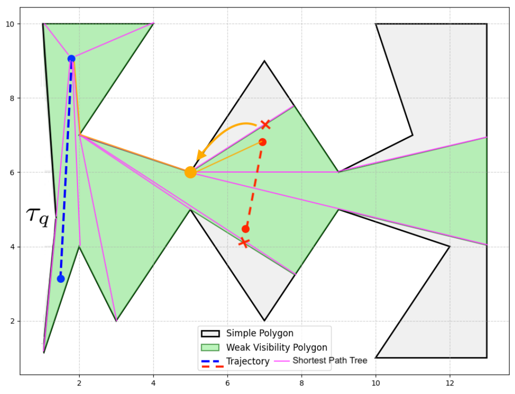
3 One Query-Provided Trajectory

In contrast to the previous section, where both trajectories are fixed, here the trajectory τ_q of entity q is provided as part of the query, while the trajectory τ_r of entity r is known a priori. The goal is to determine the first time t^* at which q and r become mutually visible within P. More specifically, suppose the trajectory $\tau_r \subset P$ of r is given. At query time, a trajectory $\tau_q \subset P$ for q is provided, as well as the query speeds v_q and v_r. For convenience, we scale the speeds by assuming that the speed of r is *one*, while the speed of q becomes $v = v_q/v_r$.

Since τ_q is provided only at the time of the query, we cannot directly construct $L(\tau_q, \tau_r)$. We instead begin with pre-computing $W(\tau_r)$ and process it to support ray-shooting queries.



(a) Given trajectory τ_q (blue) together with $T(p_q^+)$ (purple) within $W(\tau_q)$



(b) Identifying the visibility glass for the query trajectory τ_r

Figure 4 An illustration of finding the visibility glass when one trajectory is provided as a query. (a) The $T(\tau_q)$ (purple) is computed within $W(\tau_q)$. (b) This tree is used to find the boundaries of the visibility glass $L(\tau_q, \tau_r)$.

This allows us to identify only the portion of τ_q that intersects with $W(\tau_r)$ and to discard the remainder. Observe that τ_q enters $W(\tau_r)$ only once and then leaves it. That is, there will not be more than one segment on τ_q that intersects with $W(\tau_r)$. This can be verified by noting that first, the entities are not allowed to intersect P, and second, the trajectories are line segments. So, if we assume τ_q might enter $W(\tau_r)$ more than exactly once, we reach a contradiction. With a slight violation of notation, *we denote the intersection of τ_q with $W(\tau_r)$ as τ_q only during the rest of this section.*

Another step before processing a query is computing $T(p_r^+)$ and $T(p_r^-)$, restricted to vertices of $W(\tau_r)$. For each vertex $v \in T(p_r^+)$ (similarly for $T(p_r^-)$), we pre-compute the distance of v to the root, and up_j, that is a pointer to its 2^j-th ancestor on the root path, for $0 \leq j \leq \lfloor \log_2 \mathrm{depth}(v) \rfloor$.

Upon receiving a query we first locate two vertices of $W(\tau_r)$ by ray shooting: we shoot rays in the directions $\overrightarrow{p_{\tau_q}^+ p_{\tau_q}^-}$ and $\overrightarrow{p_{\tau_q}^- p_{\tau_q}^+}$, let each ray hit the boundary of $W(\tau_r)$, and choose an endpoint of the intersected edge. From these endpoints the precomputed shortest-path trees $T(p_r^+)$ and $T(p_r^-)$ immediately provide pointers into the corresponding shortest paths toward $p_{\tau_r}^+$ and $p_{\tau_r}^-$ (See Figure 4). Next, we find, on each such shortest path, the first vertex (in the path order) that is weakly visible to either $p_{\tau_q}^+$ or $p_{\tau_q}^-$. Because consecutive vertices along a shortest path are pairwise weakly visible, this vertex can be found by binary searching the path while testing weak visibility with an additional ray-shooting query. This yields pointers to $\pi(p_{\tau_q}^+, p_{\tau_r}^+)$ and $\pi(p_{\tau_q}^-, p_{\tau_r}^-)$, and hence to the visibility glass $L(\tau_q, \tau_r)$. It remains to discuss the way we eventually find t^*.

There can be two cases in this step to find t^*. Either the entities are moving in a *similar direction*, by which we mean each entity starts from p^+ (similarly for p^-) of its trajectory, or in a *different direction*, by which we mean one of them begins at p^+ and the other at p^- of its trajectory. We will suggest a general way of handling both cases.

▶ **Corollary** (2)**.** *If q's and r's positions become co-linear with a vertex $v \in \gamma^+$ (similarly for γ^-), while the line crossing q, r, and v does not intersect ∂P, obviously except at v, then a candidate for t^* is found.*

Proof. Let $v = (x_v, y_v)$, $q(t) = (x_q(t), y_q(t)) = (x_{q,0} + v_{qx}t,\ y_{q,0} + v_{qy}t)$, and $r(t) =$

$(x_r(t), y_r(t)) = (x_{r,0} + v_{rx}t,\ y_{r,0} + v_{ry}t)$, where $(x_{q,0}, y_{q,0})$ and $(x_{r,0}, y_{r,0})$ are the starting coordinates (at $t = 0$) on τ_q and τ_r respectively. The pairs (v_{qx}, v_{qy}) and (v_{rx}, v_{ry}) are the constant components of the speed vectors for q and r. For $q(t)$, v, and $r(t)$ to be co-linear, the z-component of the cross product of the vector from v to $q(t)$ and the vector from v to $r(t)$ must be zero. This can be expressed as: $(x_{q(t)} - x_v)(y_{r(t)} - y_v) - (y_{q(t)} - y_v)(x_{r(t)} - x_v) = 0$. Let $\Delta x_{q0} = x_{q,0} - x_v$, $\Delta y_{q0} = y_{q,0} - y_v$, $\Delta x_{r0} = x_{r,0} - x_v$, and $\Delta y_{r0} = y_{r,0} - y_v$. Substituting the expressions for $q(t)$ and $r(t)$: $(\Delta x_{q0} + v_{qx}t)(\Delta y_{r0} + v_{ry}t) - (\Delta y_{q0} + v_{qy}t)(\Delta x_{r0} + v_{rx}t) = 0$. Expanding this expression yields a quadratic equation in t of the form $At^2 + Bt + C = 0$, where $A = v_{qx}v_{ry} - v_{qy}v_{rx}$ [3], $B = \Delta x_{q0}v_{ry} - \Delta y_{q0}v_{rx} + \Delta y_{r0}v_{qx} - \Delta x_{r0}v_{qy}$, and $C = \Delta x_{q0}\Delta y_{r0} - \Delta y_{q0}\Delta x_{r0}$. If $A \neq 0$: $t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$, assuming $B^2 - 4AC \geq 0$ for real solutions. Otherwise, if $A = 0$ (meaning the speed vectors of q and r are parallel), there can be two cases. First, If $B \neq 0$: $t = -\frac{C}{B}$. This occurs if the initial displacement vectors are not co-linear in the same way as the speed vectors. The second is when $B = 0$. In this case, if $C = 0$, the points q_0, r_0, v are already co-linear, and since their speed vectors are parallel and related by the condition that also makes $B = 0$, they remain co-linear for all t. Otherwise, if $C \neq 0$, the initial points are not co-linear. Since the speed vectors are parallel but do not satisfy the condition for $B = 0$, no co-linearity occurs at any time t (unless by a non-physical negative t).

Every real $t \geq 0$, if found, is then a candidate for t^*, if the line crossing $q(t^*)$, $r(t^*)$, and the vertex v intersects ∂P only at that vertex. ◀

To find t^*, we perform a binary search over the vertices of *upper chain* $\gamma^+ = (v_1, \ldots, v_k)$ of $L(\tau_q, \tau_r)$ and similarly over the *lower chain* γ^-. The binary search aims to find the vertex that allows the earliest time $t \geq 0$ where $q(t)$, $r(t)$, and the vertex are collinear, and the entities become weakly visible. The binary search begins with indices *low* set to 1 and *high* set to k. It continues as long as *low* is less than *high*. In each iteration, a middle index *mid* is computed as $low + \lfloor (high - low)/2 \rfloor$. To guide this search, the earliest time that v_{mid} might allow visibility occurrence is compared to that of its successor, $v_{\text{mid}+1}$. The determination of this specific time, t_{j}, for any given vertex v_j proceeds as follows: The collinearity of $q(t), r(t)$, and v_j will be examined. If, for v_j, the coefficients A, B, and C are all zero (see Corollary 2), this vertex v_j does not define a *discrete moment* at which it establishes collinearity. Such a vertex is *discarded* from providing a t_j value. Note that if another vertex becomes co-linear with it, together with the entities, we might still be able to conclude the visibility as shown in Corollary 2. That holds if the segment connecting these four never intersects another edge or vertex of P. Otherwise, if not all of A, B, C are zero, solving $At^2 + Bt + C = 0$ yields potential non-negative times t, up to two distinct, real roots. For each such t, an $\mathcal{O}(\log n)$ ray-shooting query within P verifies if $q(t)$ and $r(t)$ are weakly visible. The t_j for this case is the minimum t from this set that also satisfies weak visibility. If no such t is found, then for comparison purposes, t_j is treated as ∞. We then compute t_{mid} and $t_{\text{mid}+1}$. If $t_{\text{mid}} \leq t_{\text{mid}+1}$, it is inferred that the vertex producing the earliest time is v_{mid} or to its left, so *high* becomes *mid*. Otherwise, *low* becomes *mid* + 1. Upon termination, the minimum time is t_{low}.

The overall t^* is the minimum of times from γ^+ and γ^-. The $\mathcal{O}(\log k)$ binary search iterations, each costing $\mathcal{O}(\log n)$ for calculating these times, yield $\mathcal{O}(\log^2 n)$ complexity per chain, as $k \in \mathcal{O}(n)$. Therefore, theorem 2 follows:

▶ **Theorem 2.** *Given the fixed trajectory $\tau_r \subset P$ of entity r, for any query-provided trajectory*

[3] Note that A is the z-component of the cross product of the speed vector of q and the speed vector of r.

$\tau_q \subset P$ and query speeds v_q and v_r, t^* can be computed in $\mathcal{O}(\log^2 n)$ time, requiring the pre-processing time of $\mathcal{O}(n \log n)$ and space of $\mathcal{O}(n \log n)$.

4 Query Trajectories

In this section, we remove all the relaxations in the previous sections, meaning none of q's and r's trajectories are known in advance, along with v_q and v_r. Before processing any query, we must pre-process P so that we can perform ray-shooting queries. To handle any constant–complexity semi-algebraic-range query [1] on the *vertices of* P as given points in $\mathbb{R}^2$, we must yet pre-process the vertices of P another time.

Once we receive τ_q and τ_r as the query, together with v_q and v_r, we consider the *area* which the *leash* between q and r may sweep in P. That is, a line segment connecting q and r that moves forward as q and r move forward in their trajectories. Denote the leash as $\ell_{qr}(t)$, and the area swept by $\ell_{qr}(t)$ as S (See Figure 5).

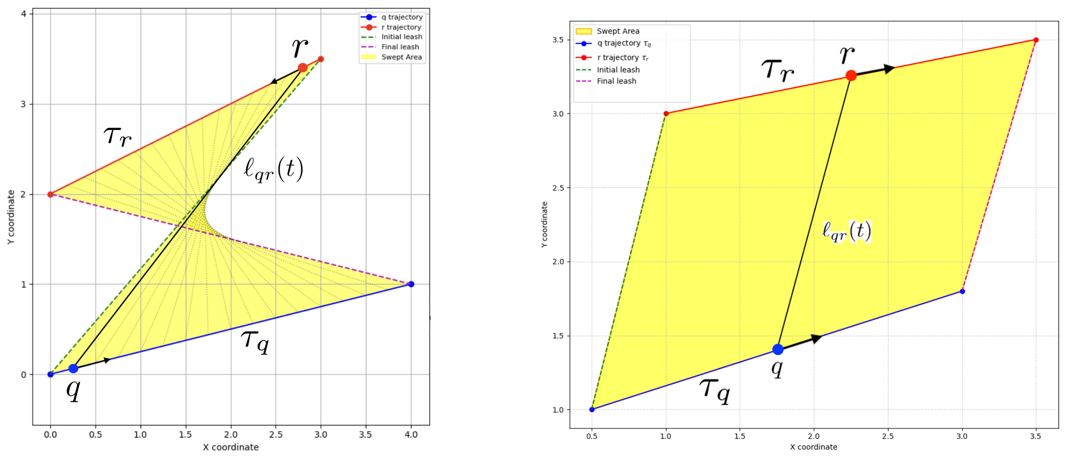
▶ **Corollary** (3). *The area swept by $\ell_{qr}(t)$ is a constant–complexity semi-algebraic range.*

Proof. Let $q(t) = (x_q(t), y_q(t))$ and $r(t) = (x_r(t), y_r(t))$. Also, let t_{qf} be the time for q to traverse τ_q, and t_{rf} be the time for r to traverse τ_r. The leash $\ell_{qr}(t)$ is defined for $t \in [0, t_f]$, where $t_f = \max(t_{qf}, t_{rf})$. The area $S = \bigcup_{t\in[0,t_f]} \ell_{qr}(t)$ swept by this leash has a boundary that includes $\ell_{qr}(0)$, $\ell_{qr}(t_f)$, τ_q, and τ_r. If the line containing $\ell_{qr}(t)$ generates an envelope, portions of this envelope to which interior points of $\ell_{qr}(t)$ are tangent can form part of the boundary of S.

The line through $q(t)$ and $r(t)$ is: $At^2+B(x,y)t+C(x,y) = 0$, where $A = v_{qx}v_{ry} - v_{qy}v_{rx}$, $B(x,y) = (v_{qy} - v_{ry})x - (v_{qx} - v_{rx})y + (x_{q0}v_{ry} + v_{qx}y_{r0} - y_{q0}v_{rx} - v_{qy}x_{r0})$, and $C(x,y) = (y_{q0} - y_{r0})x - (x_{q0} - x_{r0})y + (x_{q0}y_{r0} - y_{q0}x_{r0})$. If the speed vectors are not parallel, then $A \neq 0$, and the envelope is defined by the quadratic discriminant $B(x,y)^2 - 4AC(x,y) = 0$. Since $B(x,y)$ is linear, its square is quadratic, and $C(x,y)$ is also linear, the resulting equation describes a degree-two algebraic curve in x and y that may contribute to the boundary of region S. If the speed vectors are parallel ($A = 0$), the line becomes $B(x,y)t + C(x,y) = 0$, linear in t. In both cases, S's boundary consists of a constant number of algebraic curves, namely at most four line segments and possibly at most two degree-two curves. ◀

Clearly, one can compute S in constant time. It is also clear how to support every constant–complexity semi-algebraic-range query on the vertices of P as given points in $\mathbb{R}^2$, which includes S, namely our query of interest. So, at this stage, we perform the semi-algebraic range searching query and isolate the vertices of P that fall within S, which takes $\mathcal{O}\big(n^{1/2} \log^B n\big)$ time [1]. Once we isolate the intersecting vertices [4] of P with S, we need to find a way to reuse Corollary 2 in Section 3. Observe that since τ_q and τ_r are provided at the query time, it is not clear how one may construct $L(\tau_q, \tau_r)$ faster than $\mathcal{O}(n)$ running time. To circumvent this issue, we adopt a simple randomised approach: We uniformly at random pick a vertex among the isolated ones by the semi-algebraic range search. Same as Corollary 2, we check if q, r, and the vertex ever become co-linear. Recall that we can perform ray-shooting queries once we need to check if the segment connecting q, r, and the vertex we picked intersects any other edge or vertex of P. While we have not yet found t^*, we keep updating *low*, *mid*, and *high*, same as Section 3. As such, it is trivial to see that *in expectation*, we perform $\mathcal{O}(\log n)$ steps in our randomised binary search. Thus, we find t^* in

[4] Indeed, P is a sequence of vertices in a certain *order*. We isolate those intersecting with S.



(a) Opposite direction. **(b)** Same direction.

Figure 5 The area S (yellow) swept by the leash $\ell_{qr}(t)$ between entities q (trajectory τ_q, blue) and r (trajectory τ_r, red). The boundary of S is formed by the initial leash (green), final leash (magenta), and portions of τ_q and τ_r. An example intermediate leash $l_{qr}(t)$ is shown in black. As discussed in Corollary 3 within Section 4, S is a constant-complexity semi-algebraic range.

$\mathcal{O}(\log^2 n)$ expected time, as we perform a ray-shooting at every step of the binary search. Therefore, Theorem 3 holds:

▶ **Theorem 3.** *Given P, after $\mathcal{O}(n^{1+\varepsilon})$* expected *pre-processing time, one can find t^* once receiving τ_q and τ_r, as well as v_q and v_r, as queries, in $\mathcal{O}(n^{1/2}\log^B n)$* expected *time, where B is a constant. This requires $\mathcal{O}(n)$ space.*

5 Concluding remarks and open problems

We presented data structures for the first-visibility time t^* for two entities moving on line-segment trajectories inside a simple polygon. Two immediate open problems can be explored in the future. First, extending the results to polygonal domains with $h > 0$ holes. It is unknown whether the same strategy used in this paper can be maintained. Second, allowing one or both trajectories to be poly-lines with m segments. It is open to determine tight trade-offs between m and n for preprocessing and query costs (e.g., whether near-linear preprocessing in $n + m$ with polylogarithmic queries is achievable). Other directions include multiple-entity visibility, and letting the entities to cross ∂P (with or without the presence of holes in P).

References

1 Pankaj K Agarwal, Jiri Matousek, and Micha Sharir. On range searching with semialgebraic sets. ii. *SIAM Journal on Computing*, 42(6):2039–2062, 2013.

2 Aronov, Guibas, Teichmann, and Zhang. Visibility queries and maintenance in simple polygons. *Discrete & Computational Geometry*, 27:461–483, 2002.

3 Katarzyna Bozek, Laetitia Hebert, Yoann Portugal, Alexander S Mikheyev, and Greg J Stephens. Markerless tracking of an entire honey bee colony. *Nature communications*, 12(1):1733, 2021.

4 Clément Calenge, Stéphane Dray, and Manuela Royer-Carenzi. The concept of animals' trajectories from a data analysis perspective. *Ecological informatics*, 4(1):34–41, 2009.

5 B. Chazelle. Triangulating a simple polygon in linear time. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 270–282, 1983.

6 Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

7 David P Dobkin and Diane L Souvaine. Computational geometry in a curved world. *Algorithmica*, 5(1):421–457, 1990.

8 Somayeh Dodge, Robert Weibel, and Ehsan Forootan. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems*, 33(6):419–434, 2009.

9 Patrick Eades, Ivor van der Hoog, Maarten Löffler, and Frank Staals. Trajectory visibility. In *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.

10 Patrick Fintan Eades. *Uncertainty Models in Computational Geometry*. PhD thesis, 2020. URL: `https://hdl.handle.net/2123/23909`.

11 Mohammad Ghodsi, Anil Maheshwari, Mostafa Nouri-Baygi, Jörg-Rüdiger Sack, and Hamid Zarrabi-Zadeh. -visibility. *Computational Geometry*, 47(3, Part A):435–446, 2014. URL: `https://www.sciencedirect.com/science/article/pii/S092577211300117X`, `doi:10.1016/j.comgeo.2013.10.004`.

12 S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

13 Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. *Movement patterns in spatio-temporal data*, pages 1362–1370. Springer, Cham, 2017. 2nd edition. URL: `https://digitalcollection.zhaw.ch/handle/11475/15060`, `doi:10.1007/978-3-319-17885-1_823`.

14 Leonidas J Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.

15 Eliezer Gurarie, Russel D Andrews, and Kristin L Laidre. A novel method for identifying behavioural changes in animal movement data. *Ecology letters*, 12(5):395–408, 2009.

16 Chaowanan Jamroen, Chanon Fongkerd, Wipa Krongpha, Preecha Komkum, Alongkorn Pirayawaraporn, and Nachaya Chindakham. A novel uv sensor-based dual-axis solar tracking system: Implementation and performance analysis. *Applied Energy*, 299:117295, 2021.

17 Seyed Mohammad Hussein Kazemi, Arash Vaezi, Mohammad Ali Abam, and Mohammad Ghodsi. Trajectory range visibility. *arXiv preprint arXiv:2209.04013*, 2022.

18 D. T. Lee and F. P. Preparata. Computing visibility graphs. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 165–174, 1986.

19 Xiaojie Li, Xiang Li, Daimin Tang, and Xianrui Xu. Deriving features of traffic flow around an intersection from trajectories of vehicles. In *2010 18th International Conference on Geoinformatics*, pages 1–5. IEEE, 2010.

20 Elefterios A Melissaratos and Diane L Souvaine. Shortest paths help solve geometric optimization problems in planar regions. *SIAM Journal on Computing*, 21(4):601–638, 1992.

21 S Pane, V Iacovacci, E Sinibaldi, and A Menciassi. Real-time imaging and tracking of microrobots in tissues using ultrasound phase analysis. *Applied Physics Letters*, 118(1):014102, 2021.

22 Andreas Stohl. Computation, accuracy and applications of trajectories—a review and bibliography. *Atmospheric Environment*, 32(6):947–966, 1998.