Sweeping Simple Polygons with a Chain of Guards

Alon Efrat* Leonidas J. Guibas† Sariel Har-Peled‡ David C. Lin§
Joseph S. B. Mitchell¶ T. M. Murali‖

September 1, 1999

Abstract

We consider the problem of locating a continuously-moving target using a group of guards moving inside a simple polygon. Our guards always form a simple polygonal chain within the polygon such that consecutive guards along the chain are mutually visible. We develop algorithms that sweep such a chain of guards through a polygon to locate the target. Our two main results are the following:

1. an algorithm to compute the minimum number r^* of guards needed to sweep an n-vertex polygon that runs in $O(n^3)$ time and uses $O(n^2)$ working space, and
2. a faster algorithm, using $O(n \log n)$ time and $O(n)$ space, to compute an integer r such that $\max(r - 16, 2) \leq r^* \leq r$ and P can be swept with a chain of r guards.

We develop two other techniques to approximate r^*. Using $O(n^2)$ time and space, we show how to sweep the polygon using at most $r^* + 2$ guards. We also show that any polygon can be swept by a number of guards equal to two more than the link radius of the polygon.

As a key component of our exact algorithm, we introduce the notion of the *link diagram* of a polygon, which encodes the link distance between all pairs of points on the boundary of the polygon. We prove that the link diagram has size $\Theta(n^3)$ and can be constructed in $\Theta(n^3)$ time. We also show link diagram provides a data structure for optimal two-point link-distance queries, matching an earlier result of Arkin et al.

As a key component of our $O(n \log n)$-time approximation algorithm, we introduce the notion of the "link width" of a polygon, which may have independent interest, as it captures important structural properties of simple polygons.

*Computer Science Dept., Stanford University, 353 Serra Mall, Stanford CA 94305. Email: alon@cs.stanford.edu.
Supported by a Rothschild Fellowship and by NSF research grant CCR-96-23851.

†Computer Science Dept., Stanford University, 353 Serra Mall, Stanford CA 94305. Email: guibas@cs.stanford.edu.
Partially supported by DARPA grant DAAE07-98-C-L027 and by ARO MURI grant DAAH04-96-1-007.

‡School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel. Email: sariel@math.tau.ac.il.

§Computer Science Dept., Stanford University, 353 Serra Mall, Stanford CA 94305. Email: dlin@cs.stanford.edu.
Supported by DARPA grant DAAE07-98-C-L027 and by ARO MURI grant DAAH04-96-1-007.

¶Dept. Applied Math, University at Stony Brook, Stony Brook, NY 11794-3600. Email: jsbm@ams.sunysb.edu.
Partially supported by NSF (CCR-9732220), and grants from Hughes Research Labs, ISX Corporation, Seagull Technologies, NASA, and Sun Microsystems.

‖Computer Science Dept., Stanford University, 353 Serra Mall, Stanford CA 94305. Email: murali@cs.stanford.edu.
Supported by DARPA grant DAAE07-98-C-L027.

1 Introduction

Both visibility and motion planning questions have instigated fruitful investigations in computational geometry and given rise to well-studied areas, such as art-gallery problems [16, 23, 31], ray-shooting queries of various sorts [2, 7, 10, 25], and the combinatorics and algorithms of arrangements [1, 13, 14]. Little work, however, has been done at the interface between these two areas, where visibility becomes a tool, or a goal of motion planning. Perhaps the most classic example of such work is the computation of "watchman tours" inside a simple polygon [4, 6, 8]; a *watchman tour* of a polygon is a closed path inside the polygon such that every point of the polygon is visible from some point on the tour.

In this paper, we focus on multiple mobile guards whose motion planning goal is to explore a 2-D workspace, which in our case is a simple polygon. In this polygon, there may be one or more moving targets; nothing is known about the location of the targets or their motion abilities, except that their motion must be continuous. The goal of the guards is to "see" the targets, or to verify that no target is present in the polygon. The guards see a target when there is an unobstructed line-of-sight between it and one of the guards. We may impose various limitations on the viewing frustum and the range of the vision sensors of the guards.

Parsons [24] and Megiddo et al. [22] study a similar problem in the context of pursuit-evasion in a graph; in this scenario, the guards and target can move from vertex to vertex of a graph, until a guard and the target eventually lie in the same vertex. In our geometric setting, what makes this problem challenging is the issue of *recontamination*: a particular region of the polygon may have been cleared by the guards, but if the target can find a way to enter the region again, it becomes recontaminated and must again be cleared. Thus, unless one has sufficiently many guards, the target finding problem is not always solvable. Crass et al. [9], Suzuki and Yamashita [28], Guibas et al. [12], and LaValle et al. [21] study various versions of this problem where the guards move independently. Guibas et al. prove that for a simple polygon with n vertices and h holes, $\Theta(\sqrt{h} + \log n)$ guards are needed in the worst case to detect all targets. They also prove that computing the smallest number of guards needed to find a moving target in a polygonal environment is NP-hard.

In this paper, we look at a more constrained but still realistic model of how a polygon might be cleared by a group of guards. We assume that the guards always form a simple polygonal chain through the polygon; the guards at the ends of the chain are always on two edges of the polygon, while the rest are at internal vertices of the chain. All links in the chain are segments inside the polygon. Thus the guards are mutually visible in pairs and are all linked together. Such a guard configuration has obvious advantages for safety and communication, if this target-finding operation happens in adversarial settings. Our goal is to sweep the polygon with a continuously moving chain of guards, so that, at any instant, the chain of guards partitions the polygon into a "cleared" region and an "uncleared" region. In the end, we would like to ensure that every point of the polygon has been swept over an odd number of times. This property guarantees that if any targets are present in the polygon, they will have to be swept over by the guard chain and thus discovered.

There has been considerable work on the class of polygons that can be swept with a chain of only two observers—these polygons are called *streets* [15, 17, 20, 30]. In the framework of Icking and Klein [17], the guards are required to start at a point p on the boundary of the polygon and finish at a point q also on the boundary of the polygon. One guard moves clockwise from p to q and the other moves counterclockwise from p to q. Given p and q, Icking and Klein show how to check whether the polygon can be swept by the two guards under these constraints in $O(n \log n)$

time. If a sweep exists, they construct it in $O(n \log n + k)$ time, where k is the number of "walk" instructions given to the guards to implement the sweep. Heffernan [15] shows that $O(n)$ time suffices to check whether a sweep by two guards exists between p and q. Tseng et al. [30] consider the problem of finding two points p and q on the boundary of the polygon such that a straight walk or a straight counter-walk exists between p and q that sweeps the polygon (the guards are not allowed to backtrack in a straight walk, whereas in a straight counter-walk, one guard moves from p to q and the other from q to p without backtracking). They check if two such points exist (and output a pair) in $O(n \log n)$ time. Based on initial work by Suzuki and Yamashita [28], Tan [29] describes techniques to check in $O(n^2)$ time if a chain of two or three guards can sweep a polygon and to produce such a sweep in $O(n^3)$ time.

While these results are restricted to streets and to polygons that can be swept by three guards, we are interested in sweeping polygons that may require more than three guards. Let P be a polygon with n vertices and let r^* be the minimum number of guards needed to sweep P. Our aim is to compute r^* (or to find a good approximation to r^*) and to determine a search schedule of small complexity for the guards to perform the sweep (we formally define a search schedule and its complexity later). In this paper, we describe the following results:

1. We compute r^* in $O(n^3)$ time, using $O(n^2)$ working space, and generate a search schedule of size $O(r^* n^3)$;

2. Using $O(n^2)$ time and $O(n^2)$ space, we compute an integer $r \leq r^* + 2$ such that we can sweep P using r guards with a search schedule of size $O(rn^2)$. We can also compute in $O(n^2 r \log r)$ time a search schedule of size $O(rn^2)$ for P that uses $r + 4$ guards;

3. Using $O(n \log n)$ time and $O(n)$ space, we compute an integer r such that $r \leq r^* + 16$, and we can sweep P using r guards; and

4. We show how to sweep P using r guards, where r is two more than the link radius of P, and generate a search schedule of size $O(rn)$. (We omit the proof of this result from this abstract.)

The primary difficulty in planning motions for greater than two guards is that the guards at the internal vertices of the chain can be located anywhere in the interior of P. To solve this problem, we introduce a structure called the "link diagram" (we formally define this notion later), which represents the link distance and minimum-link paths between all pairs of points on the boundary of P. As far as we are aware, this structure appears to be a new concept. We prove that the link diagram has $\Theta(n^3)$ size and describe an algorithm to construct it in $O(n^3)$ time. In the full version of the paper, we also show how to use the link diagram to answer link-distance and minimum-link-path queries between two points in P in optimal time, matching the earlier result of Arkin et al. [5]. Our query algorithm is especially simple and avoids the case analysis of the algorithm of Arkin et al.

Our first approximation algorithm (with an additive error of two) is based on the observation that we can approximate the link diagram of P by the link distances between the $O(n^2)$ pairs of vertices of P, if we are willing to tolerate a small additive error (of at most two). Our second, more efficient, approximation algorithm (also with a small additive error) is based on an interesting relationship we establish between r^* and the "link width" of P. Suprisingly, we can show that r^* is bounded from above and from below by the link width (ignoring additive constants).

In the next section, we give some basic definitions, introduce the concept of the "link diagram," and review some facts about window partitions. The following sections describe the main results, first for exact optimization, then for approximation. Due to lack of space, we defer most proofs to the full version of the paper; some proofs are contained in the appendices.

2 Geometric Preliminaries

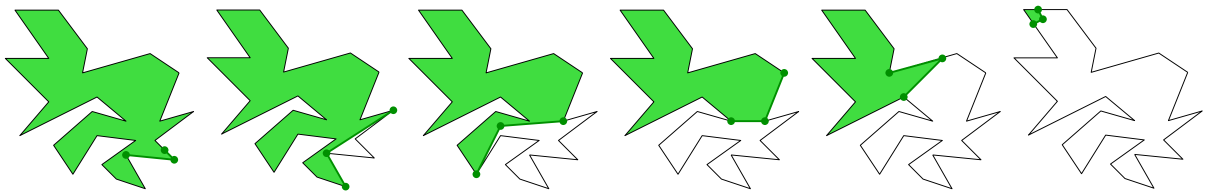
Let P be a simple polygon in the plane. Let $\mathcal{G} = \{G_1, G_2, \ldots, G_r\}$ be a set of point guards in P. For a guard $G_i \in \mathcal{G}$, let $\gamma_i(t)$ denote the position of G_i in P at time t; we require that $\gamma_i(t) : [0, \infty) \to P$ be a continuous function. A *configuration* of $\mathcal{G}$ at time t, denoted $\Gamma(t)$ is the set of points $\{\gamma_i(t) \mid 1 \leq i \leq r\}$. We say that $\Gamma(t)$ is *legal* if

1. $\gamma_1(t)$ and $\gamma_r(t)$ both lie in ∂P, and
2. for every $1 \leq i < r$, the segment $\gamma_i(t)\gamma_{i+1}(t)$ does not intersect the exterior of P.

From now on, we will use the term configuration to mean legal configuration. A useful way to think of a configuration of $\mathcal{G}$ is as a piecewise-linear path connecting the points $\gamma_1(t)$ and $\gamma_r(t)$ that "cuts" through P and does not intersect the exterior of P.

A *motion strategy* $(\gamma, \mathcal{G}) = \{\gamma_i, 1 \leq i \leq r\}$ is a specification of γ_i, for each guard $G_i \in \mathcal{G}$. We assume that each guard can follow an algebraic path, once the path is specified. Thus, each γ_i is a piecewise-algebraic function. The *complexity* of γ_i is the number of algebraic functions needed to define it. The *complexity* of a motion strategy is the total complexity of the γ_i's.

In order to formalize the notion of sweeping a polygon, we assume that the chain corresponding to the configuration of the guards is oriented from G_1 to G_r. For a motion strategy $(\gamma, \mathcal{G})$, let $A_P(t)$ denote the fraction of the area of P to the right of the configuration $\Gamma(t)$; $A_P(0) = 0$. We say that a motion strategy $(\gamma, \mathcal{G})$ is a *search schedule* for P if $A_P(t) = 1$, for some $t > 0$. Finally, we say that P is r-*searchable* if a search schedule that uses at most r guards exists for P. See Figure 1 for an example of such a sweep. In Appendix A, we show that there are n-vertex polygons that are not $o(n)$-searchable.



(a) (b) (c) (d) (e) (f)

Figure 1: A search schedule with three guards. The unswept region is shown shaded.

We assume without loss of generality that all of the guards start at the same point in ∂P at the beginning of the sweep and converge at another point of ∂P at the end of the sweep. The following lemma characterizes when a motion strategy is a search schedule:

Lemma 2.1 *Given a motion strategy* $(\gamma, \mathcal{G})$, *let* d_1 *(resp.,* d_2*) denote the total distance that* G_1 *(resp.,* G_r*) travels in the counterclockwise (resp., clockwise) direction during* γ, *divided by the perimeter of* P. *If* $|d_1 + d_2| = 1$, *then* $(\gamma, \mathcal{G})$ *is a search schedule for* P.

Using this lemma, it is easy to show that in any search schedule, each point in P is swept over an odd number of times.

In all of our algorithms, we construct search schedules where each configuration of the guards corresponds to a "minimum-link" path between the first and last guards. We now give some standard definitions related to such paths. Given two points $p, q \in P$, a *minimum-link path* between p and q is a piecewise-linear path between p and q that does not intersect the exterior of P and has the minimum number of line segments; the *link distance* $d_L(p, q)$ between p and q is the number of line segments in such a path.

We now define the link diagram of P, a structure that is central to our algorithm for computing r^*. We first select an arbitrary point $o \in \partial P$ as the origin of ∂P and parameterize every point $p \in \partial P$ by the clockwise distance from o to p along ∂P, divided by the perimeter of P. Let $f : [0, 1) \to \partial P$ denote the bijective function corresponding to this parameterization; thus, $f()$ maps every point in ∂P to a dual point in the interval $[0, 1)$. For any point (x, y) in the dual unit square, let $d_L(x, y) : [0, 1) \times [0, 1) \to \mathbb{N}$ denote the link distance between the points $f(x) \in \partial P$ and $f(y) \in \partial P$. The *link diagram* $\mathcal{L}_P$ is the graph of the function $d()$. See Figure 2 for an example of $\mathcal{L}_P$. A face of $\mathcal{L}_P$ is a maximally-connected region where the function $d()$ assumes the same value; an arc of $\mathcal{L}_P$ separates two different faces of $\mathcal{L}_P$ (the values of $d()$ in these two faces differ by 1); and a node of $\mathcal{L}_P$ is a point on the boundary of four or more faces of $\mathcal{L}_P$ or a point adjacent to two different arcs that separate the same pair of faces.[1] Note that $\mathcal{L}_P$ is symmetric since $d()$ is a symmetric function.

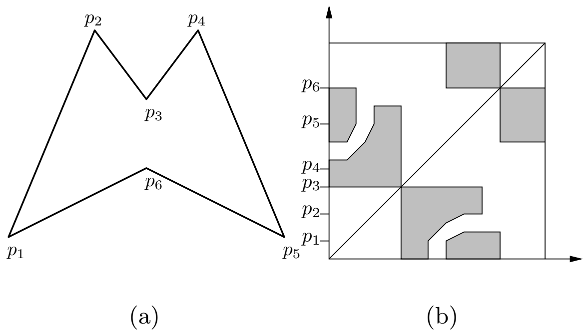


Figure 2: (a) A polygon P and (b) its link diagram. Shaded areas correspond to pairs of points on ∂P with link distance two.

Given two points $p, q \in P$, we say that p and q *see* each other if the segment pq does not intersect the exterior of P. Given two points $p, q \in P$ that see each other, let ℓ be the line passing through p and q. Then the *extension* of (p, q) is the connected component of $\ell \cap P$ that contains the segment pq.

[1]A node of $\mathcal{L}_P$ cannot be adjacent to an odd number of faces; if it is, then one of the arcs adjacent to the node separates faces where the value of $d()$ differs by zero or at least by two, which is impossible.

The *window partition* $\mathcal{W}_p$ of a $p \in P$ is a partition of P into maximal regions of constant link distance from p. An edge of $\mathcal{W}_p$ is either a portion of an edge of P or is a segment that separates two regions of $\mathcal{W}_p$; we call such a segment a *window* of $\mathcal{W}_p$. If a window $w \in \mathcal{W}_p$ has endpoints x and y, then one endpoint of w (say, x) is a reflex vertex v of P and the other endpoint (y) lies on an edge e of P; x is closer to p than y in terms of geodesic distance. We say that the *combinatorial type* of w is the vertex-edge pair (v, e). The *combinatorial type* of $\mathcal{W}_p$ is a list of the combinatorial types of all its windows. The planar dual of $\mathcal{W}_p$ is the *window tree*, $\mathcal{T}_p$. Suri [27] introduced the notion of window partition and showed that it can be constructed in time and space $O(n)$. The definitions of window partition and window tree extend naturally to the case when the source is a line segment, instead of a point.

We can use the window partition $\mathcal{W}_p$ to compute a min-link path from p to any other point in P. In general, min-link paths are not unique. The *canonical min-link path* $\pi_L(p, q)$ between $p \in \partial P$ and $q \in \partial P$ is a path that uses only extensions of windows in $\mathcal{W}_p$, with the last link chosen to pass through the last vertex of the geodesic shortest path between p and q. We define the *combinatorial type* of a link of $\pi_L(p, q)$ (except, possibly, the last link) to be the combinatorial type of the window of $\mathcal{W}_p$ of which it is an extension. Each link of $\pi_L(p, q)$ passes through a reflex vertex of P (the reflex vertex is also a vertex of the geodesic shortest path between p and q). We say that a link of $\pi_L(p, q)$ is *pinned* if it passes through two reflex vertices of P such that the vertices lie on opposite sides of the link.

Let $p = f(t)$, for some $t \in [0, 1)$, let λ be a window in $\mathcal{W}_p$ with combinatorial type (v, e), and let q be the endpoint of λ lying on e. Suppose that the canonical min-link path $\pi_L(p, q)$ from p to q does not contain any pinned edge. We can show that we can parameterize the position of q as a *homography* $q = g(t) = (A + Bt)/(C + Dt)$.

3 The Link Diagram

In this section, we prove an $O(n^3)$ bound on the size of the link diagram $\mathcal{L}_P$ of a n-vertex polygon P and describe an algorithm to construct $\mathcal{L}_P$ in $O(n^3)$ time. We also show how to compute r^* by searching $\mathcal{L}_P$ and produce a search schedule of $O(r^* n^3)$ complexity for P using r^* guards.

We first sketch the proof for bounding the size of $\mathcal{L}_P$. The first property we establish is that every vertical (or horizontal) line intersects the arcs of $\mathcal{L}_P$ at $O(n)$ points; if the line passes through the point $(t, 0)$, then these intersections correspond to the endpoints of the windows of $\mathcal{W}_{f(t)}$. We then show that if we sweep a vertical line across the plane, the line intersects nodes of $\mathcal{L}_P$ exactly at values of t such that the combinatorial type of $\mathcal{W}_{f(t)}$ changes. At each such value of t, the line intersects $O(n)$ nodes of $\mathcal{L}_P$. Arkin et al. [5] show that the combinatorial type of $\mathcal{W}_{f(t)}$ changes at $O(n^2)$ value of t. These facts imply that $\mathcal{L}_P$ has $O(n^3)$ size. An interesting implication of these arguments is that the nodes of $\mathcal{L}_P$ lie in a total of $O(n^2)$ vertical (or horizontal) lines.

Below, we describe the proof in some more detail. We first introduce some notation. Let $\ell(t)$ be the vertical line through the point $(t, 0)$ in the dual plane. Throughout this section, we will use $\varepsilon > 0$ to denote a sufficiently small real number. We will abuse notation and use $\mathcal{W}_t$, where $t \in [0, 1)$, to denote $\mathcal{W}_{f(t)}$ and use $\pi_L(t, u)$, where $t, u \in [0, 1)$, to denote $\pi_L(f(t), f(u))$. We first state a simple lemma that relates arcs of $\mathcal{L}_P$ to window partitions of points on ∂P.

Lemma 3.1 *Suppose the vertical line $\ell(t)$ does not intersect any nodes of $\mathcal{L}_P$. The line $\ell(t)$ intersects an arc of $\mathcal{L}_P$ at a point (t, u) iff $f(u)$ is the endpoint of a window of $\mathcal{W}_t$.*

Using the above lemma, it is not difficult to establish the following:

Lemma 3.2 *Let $t, u \in [0, 1)$ be such that no nodes of $\mathcal{L}_P$ are contained in the vertical strip bounded by $\ell(t)$ and $\ell(u)$. Then the combinatorial types of the window partitions $\mathcal{W}_t$ and $\mathcal{W}_u$ are identical.*

The above lemma implies that if we sweep a vertical line $\ell(t)$ across $\mathcal{L}_P$, then at every value of t such that the combinatorial types of the window partitions $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ are different, $\ell(t)$ intersects a node of $\mathcal{L}_P$. We now prove that the converse is also true, i.e., if $\ell(t)$ intersects a node of $\mathcal{L}_P$, then the combinatorial types of the window partitions $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ are different. In order to prove this fact, we first prove some more properties of the arcs and nodes of $\mathcal{L}_P$. The next two lemmas establish precise conditions for a point on an arc of $\mathcal{L}_P$ to be a node of $\mathcal{L}_P$.

Lemma 3.3 *Suppose that the point (t, u) is on an arc of $\mathcal{L}_P$ and $\pi_L(t, u)$ does not contain a pinned link. The point (t, u) is a node of $\mathcal{L}_P$ iff one of the links of $\pi_L(t, u)$ touches two vertices of P.*

Lemma 3.4 *Suppose that the point (t, u) is on an arc of $\mathcal{L}_P$ and $\pi_L(t, u)$ contains a pinned link λ. The point (t, u) is a node of $\mathcal{L}_P$ iff $f(t)$ and $f(u)$ are endpoints of a window of $\mathcal{W}_\lambda$.*

The two lemmas above have the following corollary (a window $\lambda \in \mathcal{W}_t$ divides P into two or more sub-polygons; we use $P[\lambda; f(t)]$ to denote the sub-polygons not containing $f(t)$):

Corollary 3.5 *If a window $\lambda \in \mathcal{W}_t$ touches two vertices of P, then the point (t, u) is a node of $\mathcal{L}_P$ for every value of u such that $f(u)$ is the endpoint of a window of $\mathcal{W}_\lambda$ and $f(u) \in \partial P[\lambda; f(t)]$.*

Using Lemmas 3.3 and 3.4, we can prove the following:

Lemma 3.6 *If the point (t, u) is a node of $\mathcal{L}_P$, then the window partitions $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ have different combinatorial types.*

We have now assembled all the ingredients we need to prove an $O(n^3)$ bound on the size of $\mathcal{L}_P$. We sweep the vertical line $\ell(t)$ across $\mathcal{L}_P$ from $\ell(0)$ to $\ell(1)$ and consider the intersection of $\ell(t)$ with the arcs of $\mathcal{L}_P$. Lemma 3.1 implies that this process is equivalent to moving the point $f(t)$ along ∂P and considering $\mathcal{W}_t$. Lemmas 3.2 and 3.6 imply that $\ell(t)$ intersects a node of $\mathcal{L}_P$ iff the combinatorial type of $\mathcal{W}_t$ changes. Arkin et al. [5] show that for a polygon P with n vertices, there are $O(n^2)$ values of $t \in [0, 1)$ such that $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ have different combinatorial types. Let t' be such a value of t and let λ be the window of $\mathcal{W}_{t'}$ that touches two vertices of P. Corollary 3.5 implies that the point (t', u) is a node of $\mathcal{L}_P$ only if $f(u)$ is the endpoint of a window in $\mathcal{W}_\lambda$. There are $O(n)$ such values of u. Therefore, at each of the $O(n^2)$ values of t where the combinatorial types of $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ are different, $\ell(t)$ intersects $O(n)$ nodes of $\mathcal{L}_P$. This argument proves an $O(n^3)$ bound on the size of $\mathcal{L}_P$. In Appendix A, we show that this bound is tight: there are n-vertex polygons for which $\mathcal{L}_P$ has size $\Omega(n^3)$.

Theorem 3.7 *The link diagram $\mathcal{L}_P$ of a polygon P with n vertices has size $\Theta(n^3)$.*

We now describe an algorithm to construct $\mathcal{L}_P$. The algorithm simply mimics the proof of the size bound by sweeping a vertical line $\ell(t)$ across $\mathcal{L}_P$ and maintaining the intersection of $\ell(t)$ with $\mathcal{L}_P$. We represent this intersection by a sequence $L(t)$ of $O(n)$ sorted numbers in $[0, 1)$; $u \in L(t)$ iff $f(u)$ is the endpoint of a window in $\mathcal{W}_t$. If $u \in L(t)$, we use $\sigma(t, u)$ to denote the arc of $\mathcal{L}_P$ that the point (t, u) lies on and we store the combinatorial type of $\sigma(t, u)$ with u in $L(t)$. Before describing the algorithm, we need a simple definition. Let v be a vertex of P and let p be the endpoint of a window in $\mathcal{W}_v$. If $d_L(v, p) > 1$, then the first link in $\pi_L(v, p)$ passes through v and another vertex of P. We call this link p's *source link* and denote it by s_p.

1. For each vertex $v \in P$, we compute $\mathcal{W}_v$. For every endpoint p of a window in $\mathcal{W}_v$, we compute s_p. We sort all of these endpoints around ∂P. Let Q be the sorted sequence of these endpoints.

2. We compute $L(0)$ and maintain $L(t)$ as t increases from 0 to 1. For every value of t such that $f(t)$ is a window endpoint in Q, we locate the window (with the same combinatorial type as) $s_{f(t)}$ in $L(t)$. For every value of u such that $f(u)$ is the endpoint of a window of $\mathcal{W}_{s_{f(t)}}$ and $f(u) \in \partial P[s_{f(t)}; f(t)]$, we add (t, u) as a node to $\mathcal{L}_P$ and end the arc $\sigma(t, u)$ at (t, u).

   (a) If $s_{f(t)}$ is not pinned, then for every new node (t, u) (added above), we add a new arc $\sigma(t, u)$ to $\mathcal{L}_P$. We obtain the equation of $\sigma(t, u)$ by appropriately updating the homography defining the arc $\sigma(t - \varepsilon, u)$.

   (b) If $s_{f(t)}$ is pinned, we add to $\mathcal{L}_P$ a vertical arc for each pair of new nodes that are adjacent along $\ell(t)$. For every new node (t, u), we also add a new horizontal arc $\sigma(t, u)$ to $\mathcal{L}_P$.

The correctness of the algorithm follows from Corollary 3.5. It is easy to analyze the running time of the algorithm. The first step takes $O(n^2 \log n)$ time. We execute the second step $O(n^2)$ times [5], spending $O(n)$ time per execution. Thus, we have the following theorem:

Theorem 3.8 *We can construct $\mathcal{L}_P$ in $O(n^3)$ time, using $O(n^2)$ working space.*

We now turn our attention to using $\mathcal{L}_P$ to compute the optimum number r^* of guards and a corresponding search schedule for r^* guards. Lemma 2.1 states that a motion strategy $(\gamma, \mathcal{G})$ is a search schedule if the total distance travelled by the extreme guards (measured counterclockwise for one guard and clockwise for the other) sums to the perimeter of P. To exploit this fact, we augment the diagram $\mathcal{L}_P$ by placing a translated copy of it (translated upwards by distance 1) just above it in the plane. Lemma 2.1 implies that any path from the diagonal $y = x$ in the bottom copy to the diagonal $y = x + 1$ in the top copy corresponds to a search schedule for P. Our algorithm for computing r^* is simple. We consider the graph defined by the nodes and arcs of the two copies of $\mathcal{L}_P$. We label each arc and each node with the smallest link distance associated with the faces adjacent to it. We then perform a breadth-first search in this graph to compute the smallest integer r^* such that a path exists between the two diagonals that uses only arcs and nodes with labels at most $r^* - 1$ (since a chain of $r^* - 1$ links corresponds to r^* guards). We can adapt this procedure to compute a search schedule too; details appear in the full paper. Clearly, the breadth-first search takes $O(n^3)$ time and produces a path in $\mathcal{L}_P$ that visits $O(n^3)$ nodes. To compute the search schedule, at each node of this path, we may need to update the motions of at most r^* guards, thus computing a search schedule of complexity $O(r^* n^3)$.

4 Approximation Algorithms

In this section, we describe three approximation schemes: (1) an algorithm that uses $O(n^2)$ time to compute r^* within an additive error of two, (2) an algorithm that uses $O(n \log n)$ time to compute

r^* within an additive error of at most 16, and (3) a method for sweeping P that uses at most the link radius of P (which we can compute in $O(n \log n)$ time [11]) plus two guards. Here, we give details of only the first two results; we defer presenting the link radius method (which may give slightly fewer guards than method (2), in some cases) to the full paper. We also present proofs of some lemmas in Appendix C.

4.1 A simple additive approximation method

We describe a method that computes in time $O(n^2)$ an integer r such that P can be swept using r guards and $r - 2 \leq r^*$. We can also compute in $O(n^2 r \log r)$ time a schedule of $O(n^2 r)$ commands that sweeps P using a chain of at most $r + 4$ guards.

Let $e_1, e_2, \ldots, e_n$ be the edges of P. Define an $n \times n$ matrix $\mathcal{M}$, where $\mathcal{M}_{ij}$ is an upper bound on the maximum number of guards in a min-link path connecting any point of e_i to any point of e_j; namely, $\mathcal{M}_{ij} = d_L(e_i, e_j) + 3$, where $d_L(e_i, e_j) = \min_{p \in e_i, q \in e_j} d_L(p, q)$. The matrix $\mathcal{M}$ can be computed in $O(n^2)$, by computing the link distance from e_i to all other edges in $O(n)$ time [27].

As is easily shown, $\mathcal{M}$ forms an approximation to the link diagram, $\mathcal{L}_P$, since, if p is a point on an edge $e_i \subseteq \partial P$, and q is a point on an edge $e_j \subseteq \partial P$, then $d_L(p, q)$ is between $\mathcal{M}_{ij} - 3$ and $\mathcal{M}_{ij} - 1$.

Lemma 4.1 *Let π and π' be two min-link paths, both connecting an edge f to an edge f', so that $r = d_L(f, f')$. Then, we can morph π into π' using at most $r + 3$ guards. Moreover, using at most $r + 7$ guards we can compute a morphing strategy, that issues $O(r)$ commands to guards, in $O(r \log r)$ time.*

We construct a graph G on the grid $2n \times 2n$, so that two nodes are adjacent in G iff they are vertically or horizontally adjacent in the grid. We also connect the vertices on the boundary of G to the corresponding vertices on the other side of G (i.e., we "glue" together the top side of G to the bottom side of G, and the left side of G to the right side of G). For a vertex $(i, j) \in V(G)$, we assign it weight $w(i, j) = \mathcal{M}_{1+((i-1) \bmod n), 1+((j-1) \bmod n)}$. It is easy to verify that a sweeping strategy for P can be interpreted as a path σ in G connecting $(1, 1)$ to $(1, n)$, so that the maximum weight vertex along σ has weight at most two greater thatn the number of guards needed to sweep P.

On the other hand, a path σ in G connecting $(1, 1)$ to $(1, n)$, such that the maximum weight along σ is w, can be interpreted as a sweeping strategy that requires at most w guards, by Lemma 4.1. Such a min-weight path σ in G can be computed in $O(n^2)$ time using Dijkstra's algorithm. We conclude:

Theorem 4.2 *Given a simple polygon P, one can compute in $O(n^2)$ time a number r, so that P can be swept with r guards and $r - 2 \leq r^*$. Moreover, one can compute in $O(n^2 r \log r)$ time a sweeping strategy for P using at most $r + 4$ guards, with $O(n^2 r)$ commands issued to the guards.*

Proof: The algorithm for computing r is described above. For the computation of the motion strategy, we first compute the min-weight path σ in G that connects $(1, 1)$ with $(1, n)$. Next, each edge e of σ connects two configurations $\pi = (e_i, e_j)$ and $\pi' = (e_i, e_k)$.

It is now an easy matter to compute a morphing between these two configurations by computing a middle configuration π_{mid} having one guard located on a vertex $e_j \cap e_k$ of P. Next, using the

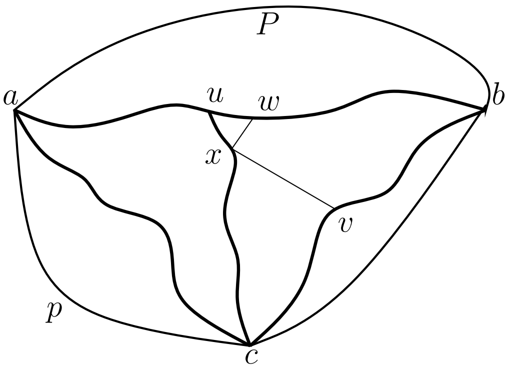


Figure 3: Definitions for Lemma 4.3

algorithm of Lemma 4.6, one can compute a morphing strategy between π and π_{mid}, and a morphing strategy between π_{mid} and π'. □

4.2 A faster additive approximation method

In this section, we describe an $O(n \log n)$-time algorithm to approximate r^* within an additive factor of 16.

For a polyline π, and any two points $p, q \in \pi$, let $a, b \in \partial P$ be a pair of points, maximizing $d_L(a, b)$; we call such a pair a *diametrical pair* of P, and let $D_P = \pi_L(a, b)$ denote a corresponding path that represents a *link diameter* of P.

We define the *link width of P relative to D_P* to be $\omega(P, D_P) = \max_{v \in P} d_L(D_P, v)$. The link width of P is then defined to be the minimum, $\min_{D_P} \omega(P, D_P)$, taken over all realizations of the diameter. (It turns out that different realizations of D_P can result in different widths, but there can be variation only by 1 link.) In our discussion, it suffices to fix one realization of the diameter, D_P, and do analysis with respect to the width $\omega = \omega(P, D_P)$. For points $p, q \in \partial P$, we let $\partial P(p, q)$ denote the portion of ∂P traced when moving from p to q in a clockwise direction (i.e., with the interior of P lying to the right). We first state two lemmas that establish the relationship between the link width and the link diameter of P.

Lemma 4.3 *Let $D_P = \pi_L(a, b)$ be a diameter of P, let c be a point that realizes the width, $\omega = d_L(c, D_P)$, and let u be a point on D_P that is closest to c in link distance. (See Figure 3.) Then, $d_L(a, u) \geq \omega - 7$ and $d_L(b, u) \geq \omega - 7$.*

Lemma 4.4 *Let $p \in \partial P(c, a)$ and $q \in \partial P(b, c)$. Then $d_L(p, b) \geq \omega - 8$, and $d_L(q, a) \geq \omega - 8$.*

Lemma 4.5 *The number of guards needed to sweep a polygon P is at least* $\max(\omega - 7, 2)$.

Proof: If there is a sweeping strategy of P by a chain of k segments ($k + 1$ guards), then it is easy to verify that during the sweep one of the following three events must happen:

- One of the guards is located at the point b and other one is located on $\partial P(c, a)$.
- One of the guards is located at the point a, and the other one is located on $\partial P(b, c)$.

- One of the guards is located at c, and the other one is located on $\partial P(a, b)$.

However, by Lemma 4.4, we know that in the first two cases $k \geq \omega - 8$. In the third case, the chain of guards must cross $\pi_L(a, b)$, which implies that $k \geq \omega$. □

Lemma 4.6 *Let $\sigma = (p_1, \ldots, p_m) \subseteq \partial P$ be a subset of ∂P that has no shortcut within P; i.e., $p_i p_{i+2} \not\subset P$. Assume that for any point $q \in \partial P$, we have $d_L(\sigma, q) \leq k$. Then, the polygon P can be swept using a chain of $k + 3$ guards.*

Proof: Let $\hat{\sigma} = \partial P \setminus \sigma$, and let $q_i \in \hat{\sigma}$ denote a point of $\hat{\sigma}$ that is closest to p_i (in link distance). Arguing as in the proof of Lemma 4.3, it follows that since σ cannot be shortcut, any point on σ sees a point of $\hat{\sigma}$; thus, $p_i q_i \subset P$. (However, note that $p_i q_i$ might cross $p_j q_j$.)

Let Q_i be the region bounded by $\partial P(q_i, q_{i+1}) || q_{i+1} p_{i+1} || p_{i+1} p_i || p_i q_i$, for $i = 1, \ldots, m - 1$. (Note that the closed curve defining Q_i may have a self-crossing at the intersection of $p_i q_i$ and $p_{i+1} q_{i+1}$.) The regions Q_i partition P. For any point $p \in \partial Q_i$, there exits a path that has at most $k + 2$ segments connecting p with p_i and that lies inside Q_i. Indeed, let $\pi = \pi_L(p, \sigma)$ be a min-link path connecting p with σ. The path π has at most k segments and must intersect (the intersection might be the endpoint of π) one of the segments $p_i q_i, p_i p_{i+1}, p_{i+1} q_{i+1}$, and thus it can be modified into a path π' that connects p with p_i that has at most $k + 2$ segments.

This implies that we can sweep Q_i in the following canonical way: (i) In the beginning the guards stand along the segment $p_i q_i$, and connect those two endpoints, (ii) In the end of the first stage of the sweep, the guards stand along the segments $p_i p_{i+1} || p_{i+1} q_{i+1}$, and (iii) In the second stage of the sweep, all of the guards standing along $p_i p_{i+1}$ are moved to stand at p_{i+1}. This sweeping requires at most $k + 3$ guards. Thus, we can sweep P by sweeping Q_1, $Q_2, \ldots$, in succession, using the above strategy. Overall, this combined strategy sweeps P using $k + 3$ guards, so that the guard who is always located on σ moves monotonically along σ. □

Theorem 4.7 $\max(\omega - 7, 2) \leq r^* \leq \omega + 5$.

Proof: Let P_1, P_2 be the two polygons formed by splitting P along $D_P = \pi_L(a, b)$. By Lemma 4.6, P_1, P_2 can be swept with $\omega + 3$ guards, so that one of the guards lies on D_P, and its movement is monotone from a towards b. Moreover, the sweeping of P_1 and P_2 is decomposed into steps where in the intermediate step only 3 guards are necessary (namely, two guards placed on an edge of the diameter, and the other guard placed on an edge of the polygon). Thus, by sweeping the regions of P_1, P_2 in an interleaving manner, we have that the number of guards necessary to sweep P is at most $\omega + 5$. The lower bound follows from Lemma 4.5. □

Theorem 4.8 *Given a polygon P, one can compute in $O(n \log n)$ time a number k, so that the number of guards needed to sweep P is between* $\max(k - 11, 2)$ *and* $k + 5$.

Proof: Compute the link-diameter, D_P, of P in $O(n \log n)$ time [18, 19, 26]. Pick a vertex v of P, and compute the window partition, $\mathcal{W}_v$, and the window tree, $\mathcal{T}_v$, in $O(n)$ time. We now mark, in linear time, all of the nodes V of $\mathcal{T}_v$ that correspond to regions of $\mathcal{W}_v$ that intersect D_P. Let μ be the vertex of $\mathcal{T}_v$ so that the minimum distance (in $\mathcal{T}_v$) to any vertex of V is maximized, and let d be this minimum distance between μ and a vertex of $\mathcal{T}_v$.

It is straightforward to verify that $\mu \leq \omega \leq \mu + 4$. Set $k = \mu + 4$. We know by Theorem 4.7, that P can be swept using $k + 5$ guards and that at least $\max(k - 11, 2)$ guards are needed. □

References

[1] P. Agarwal and M. Sharir. Arrangements. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[2] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993.

[3] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C. K. Yap. Finding minimal convex nested polygons. *Inform. Comput.*, 83(1):98–110, Oct. 1989.

[4] E. M. Arkin, J. S. B. Mitchell, and C. Piatko. Minimum-link watchman tours. Report, University at Stony Brook, 1994.

[5] E. M. Arkin, J. S. B. Mitchell, and S. Suri. Logarithmic-time link path queries in a simple polygon. *Internat. J. Comput. Geom. Appl.*, 5(4):369–395, 1995.

[6] S. Carlsson and H. Jonsson. Computing a shortest watchman path in a simple polygon in polynomial-time. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes Comput. Sci.*, pages 122–134. Springer-Verlag, 1995.

[7] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12:54–68, 1994.

[8] W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991.

[9] D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor — the open edge variant of the polygon search problem. *Internat. J. Comput. Geom. Appl.*, 5:397–412, 1995.

[10] M. de Berg. *Efficient algorithms for ray shooting and hidden surface removal.* Ph.D. dissertation, Dept. Comput. Sci., Utrecht Univ., Utrecht, Netherlands, 1992.

[11] H. N. Djidjev, A. Lingas, and J. Sack. An $O(n \log n)$ algorithm for computing the link center of a simple polygon. *Discrete Comput. Geom.*, 8(2):131–152, 1992.

[12] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit evasion in a polygonal environment. In *Proc. 5th Workshop Algorithms and Data Structures*, pages 17–30, 1997.

[13] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 21, pages 389–412. CRC Press LLC, Boca Raton, FL, 1997.

[14] D. Halperin and M. Sharir. Arrangements and their applications in robotics: Recent developments. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Proc. Workshop Algorithmic Found. Robot.*, pages 495–511. A. K. Peters, Wellesley, MA, 1995.

[15] P. J. Heffernan. An optimal algorithm for the two-guard problem. *Internat. J. Comput. Geom. Appl.*, 6:15–44, 1996.

[16] F. Hoffmann, M. Kaufmann, and K. Kriegel. The art gallery theorem for polygons with holes. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 39–48, 1991.

[17] C. Icking and R. Klein. The two guards problem. *Internat. J. Comput. Geom. Appl.*, 2(3):257–285, 1992.

[18] Y. Ke. An efficient algorithm for link-distance problems. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 69–78, 1989.

[19] Y. Ke. *Polygon visibility algorithms for weak visibility and link distance problems.* Ph.D. thesis, Dept. Comput. Sci., Johns Hopkins Univ., Baltimore, MD, 1989.

[20] R. Klein. Moving along a street. In *Proc. Computational Geometry: Methods, Algorithms and Applications*, volume 553 of *Lecture Notes Comput. Sci.*, pages 123–140. Springer-Verlag, 1991.

[21] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proc. IEEE Internat. Conf. Robot. Autom.*, Apr. 1997. To appear.

[22] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. On the complexity of searching a graph. *J. ACM*, 35:18–44, 1988.

[23] J. O'Rourke. *Art Gallery Theorems and Algorithms.* The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

[24] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, volume 642 of *Lecture Notes Math.*, pages 426–441. Springer-Verlag, Berlin, West Germany, 1976.

[25] M. Pellegrini. Ray shooting and lines in space. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 32, pages 599–614. CRC Press LLC, Boca Raton, FL, 1997.

[26] S. Suri. *Minimum link paths in polygons and related problems.* Ph.D. thesis, Dept. Comput. Sci., Johns Hopkins Univ., Baltimore, MD, 1987.

[27] S. Suri. On some link distance problems in a simple polygon. *IEEE Trans. Robot. Autom.*, 6:108–113, 1990.

[28] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21:863–888, 1992.

[29] X. Tan. Searching a simple polygon by a k-searcher. Unpublished manuscript, 1999.

[30] L. H. Tseng, P. Heffernan, and D. T. Lee. Two-guard walkability of simple polygons. *Internat. J. Comput. Geom. Appl.*, 8(1):85–116, 1998.

[31] J. Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook on Computational Geometry*, North-Holland, 1997. To appear.

A Lower Bounds

We show that there are n-vertex polygons that are not $o(n)$-searchable. Figure 4 shows such a polygon P. It consists of three "arms," L_1, L_2 and L_3, joined by a central region. Any polygonal chain lying inside P that joins a point p in the central region to the tip p_i of an arm L_i has $\Omega(n)$ segments. Suppose L_3 is the last arm to be searched in a sweep. Then, while a guard visits p_3, a guard must be positioned at a point in the central region. Otherwise, the target might escape from L_1 to L_2 or vice-versa. A similar fact holds if L_1 or L_2 is the last arm to be searched. Therefore, $\Omega(n)$ guards are needed to sweep P.

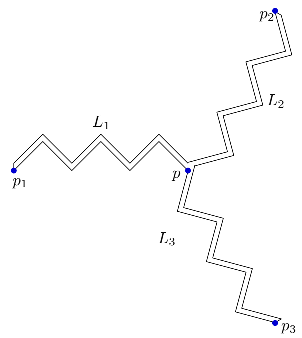


Figure 4: A polygon P such that $r^* = \Omega(n)$.

There are polygons for which the link diagram has size $\Omega(n^3)$. In Figure 5 we show a polygon P whose boundary consists of three portions: γ_1 is a convex chain of n vertices while γ_2 and γ_3 are sequences of n "teeth" each. Let $c_i, 1 \le i \le n$ denote the "base" of each tooth in γ_2 and let $d_i, 1 \le i \le n$ denote the bases in γ_3. We choose γ_1 to be small enough that every point in γ_1 can see every point of c_i and every point of d_j, for $1 \le i, j \le n$. Let c_i have endpoints p_i and q_i. Consider $\mathcal{W}_{p_i}$. Since p_i can see every point on γ_1, a window of $\mathcal{W}_{p_i}$ (in fact, a chord of the visibility polygon V_{p_i}) has an endpoint p' in ∂P to the left of the vertices of γ_1. For every $j, 1 \le i \le n$, there is a window w' in $\mathcal{W}_{p_i}$ such that w' has an endpoint $q \in d_j$. By Lemma B.3, the point $(f^{-1}(p_i), f^{-1}(q))$ is on an arc of $\mathcal{L}_P$. Now consider moving a point p from p_i to q_i. This motion causes p' to move clockwise along γ_1 and q to move clockwise along d_j. Every time p' passes a vertex of γ_1, the homography defining the motion of q (with respect to p) changes. Therefore, by the time p reaches q_i, the point $(f^{-1}(p), f^{-1}(q))$ has traced $\Omega(n)$ arcs of $\mathcal{L}_P$. The same process can be repeated for every c_i and d_j, $1 \le i, j \le n$, which implies that $\mathcal{L}_P$ has size $\Omega(n^3)$.

B Lemmas and Proofs for Section 3

Lemma B.1 *If p and q are two points in ∂P and no link of $\pi_L(p, q)$ is pinned, then the endpoint of every link of $\pi_L(p, q)$ lies on ∂P.*

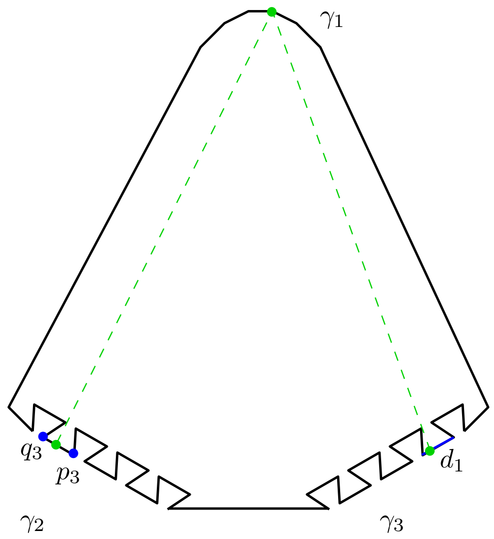


Figure 5: A lower bound construction for the size of $\mathcal{L}_P$.

Proof: We prove the lemma by contradiction. Let λ be a link of $\pi_L(p,q)$ whose endpoint does not lie on ∂P; clearly, λ is not the last link in $\pi_L(p,q)$ since q is the endpoint of the last link. Let λ' be the link following λ in $\pi_L(p,q)$; λ' is the extension of a chord of V_λ. It is well-known that the extension a chord s' of V_λ intersects λ at a point in the interior of λ iff s' passes through reflex vertices of P that lie on opposite sides of s', which implies that λ' is pinned. □

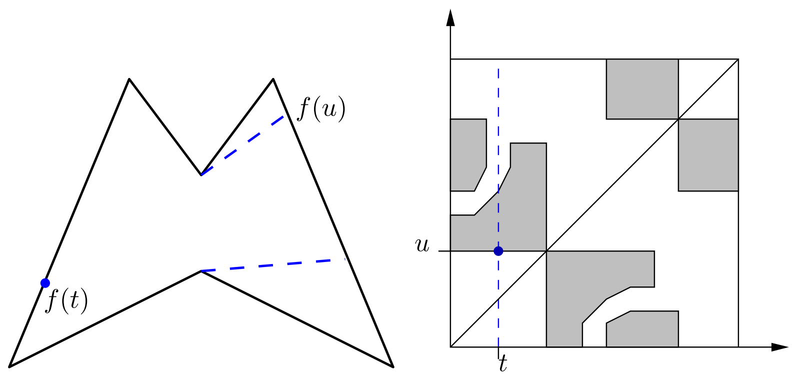
Lemma B.2 *Suppose the vertical line $\ell(t)$ through the point $(t,0)$ does not intersect any nodes of $\mathcal{L}_P$. The line $\ell(t)$ intersects an arc of $\mathcal{L}_P$ at a point (t,u) iff $f(u)$ is the endpoint of a window of $\mathcal{W}_t$.*

Proof: Consider the point $p(u) = (t,u)$ for $t \le u < 1$ and $0 \le u < t$. As u increases from t to 1 and from 0 to t, $p(u)$ moves along $\ell(t)$. Simultaneously, $f(u)$ starts at $f(t)$ and moves along ∂P back to $f(t)$. Every time $p(u)$ crosses an arc of $\mathcal{L}_P$, the link distance between $f(t)$ and $f(u)$ changes. At each such instant, $f(u)$ must lie on the endpoint of a window of $\mathcal{W}_t$, which proves the lemma. See Figure 6. □

Lemma B.3 *A point (t,u) lies on an arc of $\mathcal{L}_P$ iff $f(u)$ is the endpoint of a window of $\mathcal{W}_t$ or $f(t)$ is the endpoint of a window of $\mathcal{W}_u$.*

Proof: If $f(u)$ is the endpoint of a window of $\mathcal{W}_t$, then $d_L(t,u-\varepsilon)$ is one more or one less than $d_L(t,u+\varepsilon)$. Therefore, the points $(t,u-\varepsilon)$ and $(t,u+\varepsilon)$ lie in different faces of $\mathcal{L}_P$, which implies that (t,u) lies on an arc of $\mathcal{L}_P$. A similar proof holds if $f(t)$ is the endpoint of a window of $\mathcal{W}_u$.

We now prove the "only if" part of the lemma. We first assume that the arc (t,u) lies on is not vertical. Consider the points $p(u-\varepsilon)$ and $p(u+\varepsilon)$ for sufficiently small ε. See Figure 6. Since these two points lie in adjacent faces of $\mathcal{L}_P$, $d_L(t,u-\varepsilon)$ must differ by one from $d_L(t,u+\varepsilon)$. Therefore, the points $f(u-\varepsilon)$ and $f(u+\varepsilon)$ on ∂P must lie in different faces of $\mathcal{W}_t$, which implies that $f(u)$



(a) (b)

Figure 6: The vertical line $\ell(t)$ intersects $\mathcal{L}_P$ at (t, u) and $f(u)$ is the endpoint of a chord of $\mathcal{W}_t$.

is the endpoint of a window of $\mathcal{W}_t$. If the point (t, u) lies on a vertical arc of $\mathcal{L}_P$, we can similarly show that $f(t)$ is the endpoint of a window of $\mathcal{W}_u$. □

Remark: In the above lemma, if $f(u)$ is the endpoint of a window of $\mathcal{W}_t$ *and* $f(t)$ is the endpoint of a window of $\mathcal{W}_u$, then (t, u) lies on an arc of $\mathcal{L}_P$ that is not axis-aligned. If only one of the conditions hold, then the arc is either horizontal or vertical.

Lemma B.4 *Let $t, u \in [0, 1)$ be such that no nodes of $\mathcal{L}_P$ are contained in the vertical strip bounded by $\ell(t)$ and $\ell(u)$. Then the combinatorial types of the window partitions $\mathcal{W}_t$ and $\mathcal{W}_u$ are identical.*

Proof: We first prove that for every window w with combinatorial type (v, e) in $\mathcal{W}_t$, there is a window with combinatorial type (v, e) in $\mathcal{W}_u$. Let y be the endpoint of w that lies in e. Lemma B.3 implies that the dual point $(t, f^{-1}(y))$ lies on an arc σ of $\mathcal{L}_P$. Since there are no nodes of $\mathcal{L}_P$ in the vertical strip bounded by $\ell(t)$ and $\ell(u)$, σ also crosses $\ell(u)$. Therefore, $\mathcal{W}_u$ also has a window with combinatorial type (v, e).

By a similar argument, for every window with combinatorial type (v', e') in $\mathcal{W}_u$, there is a window with combinatorial type (v', e') in $\mathcal{W}_t$. Therefore, the combinatorial types of $\mathcal{W}_t$ and $\mathcal{W}_u$ are identical. □

Lemma B.5 *Suppose that the point (t, u) is on an arc of $\mathcal{L}_P$ and $\pi_L(t, u)$ does not contain a pinned link. The point (t, u) is a node of $\mathcal{L}_P$ iff one of the links of $\pi_L(t, u)$ touches two vertices of P.*

Proof: Let $d_L(t, u) = k$. We use $\lambda_i(t, u), 1 \le i \le k$ to denote the ith link in $\pi_L(t, u)$. Let the combinatorial type of $\lambda_i(t, u)$ be (v_i, e_i). Suppose the link $\lambda_j(t, u)$ for some $1 \le j \le k$ touches another vertex $v \in P$. We assume without loss of generality that $\lambda_j(t, u - \varepsilon)$ (the jth link in $\pi_L(t, u - \varepsilon)$) does not touch v. We consider two cases.

1. If v is an endpoint of e_i, let e' be the other edge of P incident on v. Then the combinatorial type of $\lambda_j(t, u + \varepsilon)$ (the jth link in $\pi_L(t, fu + \varepsilon)$) is (v_j, e').

2. If v is not an endpoint of e_i, then v is a reflex vertex of P that lies in the interior of $\lambda_j(t, u)$. Since there is no pinned link in $\pi_L(t, u)$, both v_j and v lie on the same side of $\lambda_j(t, u)$. As a result, the combinatorial type of $\lambda_j(t, u + \varepsilon)$ is (v, e_j).

In both cases, the homography representing the position of $\lambda_j(t, u - \varepsilon)$'s endpoint is different from the homography representing the position of $\lambda_j(t, u + \varepsilon)$'s endpoint, which implies that the homography representing the position of λ_k's endpoint changes at t. Therefore, (t, u) is a node of $\mathcal{L}_P$.

We now prove the "only if" part of the lemma. Suppose the point (t, u) is a node of $\mathcal{L}_P$. If no link of $\pi_L(t, u)$ touches two vertices of P, then $\pi_L(t, u - \varepsilon)$, $\pi_L(t, u)$, and $\pi_L(t, u + \varepsilon)$ are combinatorially identical. Therefore, the homographies representing the endpoints of the links in these three paths are the same, which contradicts the fact that (t, u) is a node of $\mathcal{L}_P$.

□

Lemma B.6 *Suppose that the point (t, u) is on an arc of $\mathcal{L}_P$ and $\pi_L(t, u)$ contains a pinned link λ. The point (t, u) is a node of $\mathcal{L}_P$ iff $f(t)$ and $f(u)$ are endpoints of a window of $\mathcal{W}_\lambda$.*

Proof: Let σ be the arc of $\mathcal{L}_P$ that (t, u) lies on. Let λ be the first pinned link on $\pi_L(t, u)$. Since λ is a link on $\pi_L(t, u)$, λ is the extension of a window $w \in \mathcal{W}_t$. We can show that σ is either vertical or horizontal.

We first prove the "if" part of the lemma. We first consider the case when σ is vertical. Since $f(u)$ is an endpoint of a window of $\mathcal{W}_\lambda$, $f(u)$ is also the endpoint of a window of $\mathcal{W}_t$. Therefore, $d_L(t, u - \varepsilon)$ differs by one from $d_L(t, u + \varepsilon)$, which implies that (t, u) is a node of $\mathcal{L}_P$. Similarly, if σ is horizontal, we have that (t, u) is a node of $\mathcal{L}_P$ because $f(t)$ is an endpoint of a window of $\mathcal{W}_\lambda$.

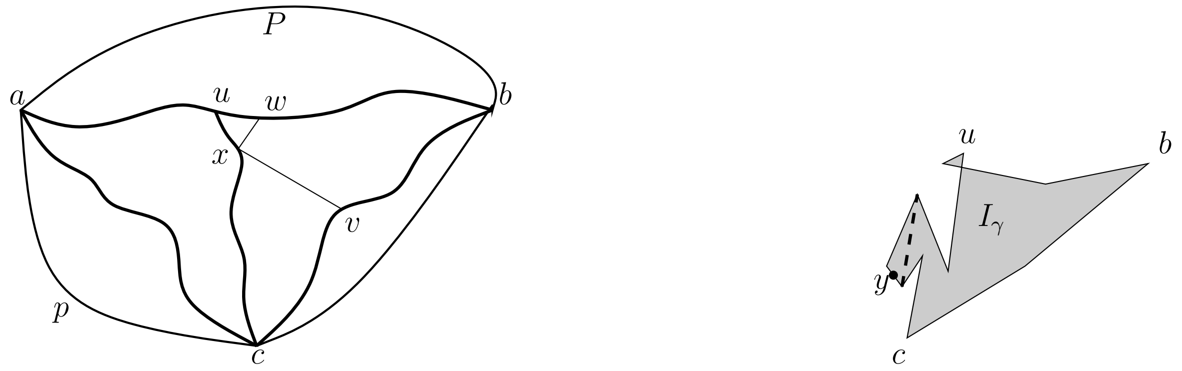
We now prove the "only if" part of the lemma. We know that (t, u) is a node of $\mathcal{L}_P$. Therefore, $d_L(t, u - \varepsilon) \neq d_L(t, u + \varepsilon)$, which implies that $f(u)$ is the endpoint of a window of $\mathcal{W}_t$. Since λ is a link on $\pi_L(t, u)$, we have that $f(u)$ is the endpoint of a window of $\mathcal{W}_\lambda$.

To show that $f(t)$ is the endpoint of a window of $\mathcal{W}_\lambda$, we first note that $f(t)$ is the endpoint of a window of $\mathcal{W}_u$. Now, λ is a link on $\pi_L(u, t)$. Therefore, $f(t)$ is the endpoint of a window of $\mathcal{W}_\lambda$. □

Lemma B.7 *If the point (t, u) is a node of $\mathcal{L}_P$, then the window partitions $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ have different combinatorial types.*

Proof: We consider two cases. Lemma B.5 implies that there is a link $\lambda \in \pi_L(t, u)$ that touches two vertices of P. Let (v, e) be the combinatorial type of λ in $\mathcal{W}_{t-\varepsilon}$ and let v' be the other vertex that λ touches in $\mathcal{W}_t$. If v' is an endpoint of e and e' is the other edge of P incident on v', then there is a window with combinatorial type (v, e') in $\mathcal{W}_{t+\varepsilon}$. If v' is not an endpoint of e, then there is a window with combinatorial type (v', e) in $\mathcal{W}_{t+\varepsilon}$. In either case, $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ have different combinatorial types.

Now we consider the case when $\pi_L(t, u)$ contains a pinned link λ. Lemma B.6 implies that $f(t)$ is the endpoint of a window of $\mathcal{W}_\lambda$. If $\mathcal{W}_{t-\varepsilon}$ and $\mathcal{W}_{t+\varepsilon}$ have the same combinatorial types, then $d_L(t - \varepsilon, \lambda) = d_L(t + \varepsilon, \lambda)$, which contradicts the fact that $f(t)$ is the endpoint of a window of $\mathcal{W}_\lambda$. □



(a) definitions (b) shortcut is possible.

Figure 7: Lemma 4.3

C Lemmas and Proofs for Section 4

C.1 A simple additive approximation method

Lemma C.1 *Let π and π' be two min-link paths, both connecting an edge f to an edge f', so that $r = d_L(f, f')$. Then, we can morph π into π' using at most $r + 3$ guards. Moreover, using at most $r + 7$ guards we can compute a morphing strategy, that issues $O(r)$ commands to guards, in $O(r \log r)$ time.*

Proof: Note that the min-link distance between any point of f and any point of f' is at most $r + 2$, thus the guards can continously move between f and f' using at most $r + 3$ guards. Unfortunately, computing this sweeping strategy requires the link diagram of P, which is too expensive to compute.

Alternatively, we now sketch an algorithm to compute a strategy that uses at most $r + 7$ guards. Let γ be the closed connected curve made out of π, π' and the relevant portions of f and f', so that $\pi, \pi' \subseteq \gamma$ (note that γ might have self-intersections). Let I_γ denote the interior of the region delimited by γ, and let $I_1, \ldots, I_k$ be the connected components of the interior of I_γ.

We deform between $\pi_i = \pi \cap \partial I_i$ and $\pi'_i = \pi' \cap \partial I_i$, so the motion is restricted to lie inside I_i, for $i = 1, \ldots, k$. To do so, we place two extra guards at the common endpoints of π_i and π'_i. Placing those two extra guards, might require moving one (extra) guard from one endpoint of π_i to the other endpoint of π_{i+1}, and this can be done by issuing a linear number of commands to the guards.

Since π and π' are both min-link paths, we know that the number of guards in π_i and π'_i is the same, up to at most an additive error of 2. Thus, we triangulate the polygon I_i, and use the chords of the triangulation to perform the continuous motion from π_i to π'_i. In the end of this continuous motion, we now move all of the extra guards placed along π'_i to its common endpoint with π'_{i+1} (since the number of those extra guards is at most 4, this requires a linear number of commands). Similarly, we move all guards that lie on the middle of an edge to the this endpoint.

We continue in this manner, until we have deformed π into π'. This motion required at most $r + 7$ guards, and a linear number of commands, and it can be computed in $O(r \log r)$ time. $\square$

C.2 A faster additive approximation method

Lemma C.2 *Let $D_P = \pi_L(a,b)$ be a diameter of P, let c be a point that realizes the width, $\omega = d_L(c, D_P)$, and let u be a point on D_P that is closest to c in link distance. (See Figure 7(a).) Then, $d_L(a,u) \geq \omega - 7$ and $d_L(b,u) \geq \omega - 7$.*

Proof: Note that we can assume, without loss of generality, that $\pi_L(a,b)$ and $\pi_L(c,u)$ do not intersect in their interior. Let γ be the curve $\pi_L(c,u)||\pi_L(u,b)||\pi_L(b,c)$, where $||$ denotes the concatenation operator. The curve γ is a closed curve, and it might be self intersecting. Let I_γ denote the region delimited by γ (i.e., the union of bounded faces in the arrangement induced by γ).

Observe that any point y of $\pi_L(c,u)$ can be connected to a point either of $\pi_L(u,b)$ or of $\pi_L(b,c)$ by a segment that does not intersect those polygonal paths in their interior. Indeed, if this is not so, then there exists a point $y \in \pi_L(c,u)$, such that any ray emanating from y directed into I_γ hits $\pi_L(c,u)$; see Figure 7(b). In particular, in any triangulation of I_γ, the triangle T that contains y must have all its vertices on $\pi_L(c,u)$. implying that it is possible to shortcut $\pi_L(c,u)$, using the edge of T that does not belong to $\pi_L(c,u)$. However, this contradicts the minimality (in the link distance) of $\pi_L(c,u)$.

This implies that there is a point $x \in \pi_L(c,u)$ that "sees" both $\pi_L(b,c)$ and $\pi_L(b,u)$; namely, there are two points v, w on $\pi_L(b,c)$ and $\pi_L(b,u)$, respectively, so that the segments xv and xw do not intersect γ in their interior. We have the following inequalities (all follow from the triangle inequality for the min-link distance):

- $d_L(c,x) - 1 \leq d_L(c,v) \leq d_L(c,x) + 1$.
- $d_L(c,x) \leq \omega = d_L(c,u) \leq d_L(c,x) + 1$
- $d_L(b,w) - 2 \leq d_L(b,v) \leq d_L(b,w) + 2$.
- $d_L(c,v) + d_L(b,v) - 1 \leq d_L(b,c)$.

These inequalities imply that $d_L(c,x) - 1 + d_L(b,w) - 2 - 1 \leq d_L(b,c)$. Hence,

$$d_L(c,u) - 1 - 4 + d_L(b,u) - 2 \leq d_L(b,c) \leq d_L(a,b) \leq d_L(a,u) + d_L(b,u),$$

using the fact that $d_L(a,b)$ is the diameter of P. We conclude that $\omega - 7 = d_L(c,u) - 7 \leq d_L(a,u)$, and, by symmetry, that $\omega - 7 \leq d_L(b,u)$. □

Lemma C.3 *Let $p \in \partial P(c,a)$ and $q \in \partial P(b,c)$. Then $d_L(p,b) \geq \omega - 8$, and $d_L(q,a) \geq \omega - 8$.*

Proof: We prove that $d_L(p,b) \geq \omega - 8$; the second inequality is shown symmetrically. We may assume that u is chosen to be closest to b along $\pi_L(a,b)$ among all choices of u that realize the link width.

We claim that the path $\pi_L(p,b)$ must intersect the visibility polygon, V_u. This will suffice to prove the lemma, since it implies that $d_L(b,u) \leq d_L(p,b) + 1$ (since, once the path $\pi_L(b,p)$ enters V_u, one additional link suffices to reach u), which implies that $d_L(p,b) \geq \omega - 8$ (since Lemma C.2 says that $\omega - 7 \leq d_L(b,u)$).

If, to the contrary, $\pi_L(p,b)$ does not intersect V_u, then the points b and p must lie in the same pocket of V_u, separated from u by a window, rr'. Since a and b are in different pockets of V_u, it follows that c lies in the same pocket as p and b. Both paths $\pi_L(b,u)$ and $\pi_L(c,u)$ must cross the window rr'. This implies that there is a path of link length $d_L(c,u)$ that joins c to a point, $u' \in rr'$, of $\pi_L(b,u)$ that is closer to b than u, contradicting our choice of u. □