New Similarity Measures between Polylines with Applications to Morphing and Polygon Sweeping*

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

April 10, 2007

Abstract

We introduce two new related metrics, the *geodesic width* and the *link width*, for measuring the "distance" between two non-intersecting polylines in the plane. If the two polylines have n vertices in total, we present algorithms to compute the geodesic width of the two polylines in $O(n^2 \log n)$ time using $O(n^2)$ space and the link width in $O(n^3 \log n)$ time using $O(n^2)$ working space `where` n `is the total number of edges of the polylines`. Our computation of these metrics relies on two closely-related combinatorial strutures: the *shortest-path diagram* and the *link diagram* of a simple polygon. The shortest-path (resp., link) diagram encodes the Euclidean (resp., link) shortest path distance between all pairs of points on the boundary of the polygon. We use these algorithms to solve two problems:

- Compute a continuous transformation that "morphs" one polyline into another polyline. Our morphing strategies ensure that each point on a polyline moves

*Preliminary versions of this paper appeared in the *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms* [EGH+00] and the *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms* [EGHM01]

†Address: Department of Computer Science, University of Arizona, Tucson AZ 85721-0077. Email: `alon@cs.arizona.edu`. WWW: `http://www.cs.arizona.edu/people/alon/`. The author did part of this research while affiliated with Stanford University.

‡Address: Computer Science Dept., Stanford University, Stanford CA 94305. Email: `guibas@cs.stanford.edu`. WWW: `http://graphics.stanford.edu/~guibas`. Partially supported by NSF (CCR-9910633), by U.S. Army Research Office MURI grant DAAH04-96-1-007, and a grant from the Intel Corporation.

§Address: Dept. of Computer Science, University of Illinois, Urbana, IL 61801-2987. Email: `sariel@cs.uiuc.edu`. WWW: `http://valis.cs.uiuc.edu/~sariel/`. The author did part of this research while affiliated with Duke University.

¶Address: Applied Mathematics and Statistics, University at Stony Brook, Stony Brook, NY 11794-3600. Email: `jsbm@ams.sunysb.edu`. WWW: `http://www.ams.sunysb.edu/~jsbm`. Partially supported by NSF (CCR-9732220), a DARPA subcontract from HRL Laboratories, NASA Ames Research (NAG2-1325), Northrop-Grumman Corporation, and Sun Microsystems.

‖Address: Bioinformatics Program, Boston University, Boston MA 02215. Email: `murali@bu.edu`, WWW: `http://people.bu.edu/murali`. The author did part of this research while affiliated with Stanford University.

as little as necessary during the morphing, that every intermediate polyline is also simple `and disjoint to any other intermediate polyline`, and that no portion of the polylines to be morphed is stretched or compressed by more than a user-defined parameter during the entire morphing. We present an algorithm that computes the geodesic width of the two polylines and utilizes it to construct a corresponding morphing strategy in $O(n^2 \log^2 n)$ time using $O(n^2)$ space. We also give an $O(n \log n)$ time algorithm to compute a 2-approximation of the geodesic width and a corresponding morphing scheme.

- Locate a continuously-moving target using a group of guards moving inside a simple polygon. The guards always determine a simple polygonal chain within the polygon, with consecutive guards along the chain being mutually visible. We compute a strategy that sweeps such a chain of guards through the polygon in order to locate a target. We compute in $O(n^3)$ time and $O(n^2)$ working space the minimum number r^* of guards needed to sweep an n-vertex polygon. We also give an approximation 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.

1 Introduction

A basic question that arises when comparing shapes (point sets, polygons, images, triangular meshes, molecules, etc.) is that of defining a metric that measures the difference between the two shapes. Depending on the application, a well-defined metric will capture one's intuitive notion of similarity while being mathematically precise and efficiently computable. The Hausdorff metric is a famous example of a metric for point sets and images [HKR93]. One of the first metrics defined to measure the difference between two polylines in the plane was the Fréchet metric. An intuitive way to understand this metric is as follows: Let α and β be the two polylines. Imagine that a man walks from one end of α to the other end and that a dog walks from one end of β to the other end with the man holding the dog by a leash. The man and the dog must both walk continuously and their motion is required to be monotonic. The Fréchet distance between α and β is the minimum leash length needed. Formally, let $d(p, q)$ denote the Euclidean distance between two points p and q in the plane. The Fréchet distance between α and β is

$$\mathcal{F}(\alpha, \beta) = \min_{f:[0,1]\to\alpha, g:[0,1]\to\beta} \max_t d(f(t), g(t)),$$

where f and g are continuous nondecreasing functions defining the positions of the man and the dog on the curve at every instant.

In this paper, we introduce two new metrics for measuring the distance between two polylines and use them to solve problems motivated by applications in computer graphics and robotics. Our metrics are motivated by the observation that in the Fréchet metric, the leash is allowed to cross the two polylines. A natural restriction to apply is to require that the leash not cross the polylines. We have two standard ways in which to measure the length of the leash: its Euclidean length and its link length (the number of line segments comprising it). Each of these measures directly leads to the new polyline metrics we define:

1. the *geodesic width*

$$W(\alpha, \beta) = \min_{f:[0,1]\to\alpha, g:[0,1]\to\beta} \max_t d_E(f(t), g(t)),$$

where $d_E(p, q)$ denotes the length of a shortest path between p and q that does not cross α and β, and lies between the two shortest paths connecting the endpoints of α and β. The minimization is over continuous nondecreasing functions. There are four "reasonable" ways to connect the endpoints of α and β, and $W(\alpha, \beta)$ is defined as the minimum obtained among all four.

2. the *link width*

$$\mathcal{L}(\alpha, \beta) = \min_{f:[0,1]\to\alpha, g:[0,1]\to\beta} \max_t d_L(f(t), g(t)),$$

where $d_L(p, q)$ denotes the minimum number of edges in a piecewise-linear path between p and q that does not cross α and β, and changes continuously as t grows from 0 to 1. Here we do not require that f and g are nondecreasing.

Each of these metrics has a natural interpretation. Consider the Euclidean shortest path (geodesic) $\pi(t)$ between $f(t)$ and $g(t)$ that does not cross α and β. If we let every point $f(t), 0 \le t \le 1$ move along $\pi(t)$ at a velocity proportional to the length $d_E(f(t), g(t))$ of $\pi(t)$ we obtain a continuous sequence of polylines. When we view these polylines in sequence, we see an animation that morphs α into β. Since geodesics do not cross (although they may coincide), We have the useful and desirable property that no intermediate polyline intersects itself, and no two intermediate polylines cross each other.

In the case of the link width, we consider a minimum-link path between $f(t)$ and $g(t)$, as t increases from 0 to 1, yielding a sequence of polylines that "sweeps" the area between α and β. If we imagine that the vertices of these polylines correspond to guards and that the edges of these polylines correspond to lines of sight between two adjacent (along the sweeping polyline) guards, then the sweep corresponds to a motion of a set of guards that can find any continuously-moving intruder in the area between α and β [1]. We also have the property that the number of guards needed is as small as possible.

The solutions to both of these problems is similar in spirit, sharing many main ideas. In the rest of this section, we discuss the morphing and sweeping problems in more detail.

1.1 Morphing Polylines

In the last few years, the problem of continuously morphing or deforming an object or image into another object or image has become increasingly popular in computer graphics and computer vision. In this problem, we are given two shapes α and β and we are asked to produce a continuous sequence of shapes "between" α and β. Rendering this sequence continuously as an animation will show α transforming into β. This problem has applications in animation, special effects in movies and entertainment, contour interpolation in medical imaging, computer-aided geometric design, and handwriting recognition.

There are many qualities that are desirable in a good morphing scheme. Since α and β are usually connected and simple, all intermediate shapes should also be connected and

[1] We ignore issues related to the intruder going "around" α and β and creeping up behind the guards.

simple. The morphing should transform a connected portion of α to a connected portion of β. It is also useful for the transformation to treat α and β as near-rigid objects and to avoid superfluous deformations during the morphing.

There are two common ways used in the literature to specify morphing schemes. The first approach uses zero sets of implicit functions to represent the morphing [CSL98, HWK94, Hug92, KR91]. The zero sets of two implicit functions represent α and β. Interpolating between (the zero sets of) these functions produces the morphing. Turk and O' Brien [TO99] combine the problems of creating and interpolating implicit functions by casting the problem in a dimension one more than the dimension in which α and β are embedded. These techniques are elegant and have proven to be quite successful. However, they have the drawback that unless the implicit functions are chosen with great care, intermediate shapes are not guaranteed to be simple. Another problem with these methods is that the correspondences between various parts of α and β are implicit and not user-controllable.

The second popular approach first computes correspondences between points of α and β (for instance, the vertices of α and β, if α and β are polyhedral) and then creates intermediate shapes as defined by interpolated positions between corresponding points [GSL$^+$99, KCP92, SGWM93, SG92]. Since the interpolated positions are usually chosen to lie on the segments or splines joining corresponding points, these techniques usually make it very difficult to ensure that intermediate shapes do not self-intersect.

Recently, Surazhsky and Gotsman [GS01, SG01a, SG01b] have proposed elegant methods for computing a morphing between polygons, such that no intermediate polygon intersects itself; however, intermediate polygons (as opposed to polylines) may intersect each other, so our goals for polylines cannot be met directly by their polygon morphing schemes. Also, there is no guarantee that these schemes minimize the amount of change needed during the morphing.

Surprisingly, the problem of computing continuous morphings has so far received relatively little attention in the theoretical computational geometry community. If α and β are n-vertex *parallel* polygons, i.e., polygons with the same sequence of angles, Hershberger and Suri [HS95], improving upon an earlier result of Guibas and Hershberger [GH94], showed that α and β can be morphed into one another such that every interpolating polygon is also parallel to α and β using $O(n \log n)$ moves; roughly speaking, a parallel move is a translation of a side of a polygon parallel to itself. When α and β are simple polygons, an approach that has been used is to find "compatible" decompositions of α and β and to use these decompositions to generate the correspondences between α and β. Aronov et al. [ASS93] show that a compatible triangulation (also called a piecewise-linear homeomorphism) of size $\Theta(n^2)$ exists between two n-vertex simple polygons. Gupta and Wenger [GW97] construct compatible triangulations whose size is within a constant factor of optimal. Shapira and Rappaport [SR95] decompose the polygons into star-shaped pieces. Their technique is not able to avoid all self-intersections during the morphing and can take $O(n^4)$ time.

In the first part of this paper, we consider the problem of morphing two non-intersecting simple polygonal chains (or polylines) α and β in the plane. A *morphing scheme* $\Gamma(\alpha, \beta) = \{\gamma(t) \mid 0 \leq t \leq 1\}$ from α to β is a family of polylines such that $\alpha = \gamma(0), \beta = \gamma(1)$, for every $0 \leq t \leq 1$, $\gamma(t)$ is connected and simple, and the scheme is *continuous*, meaning that for any t and any $\varepsilon > 0$, there is a neighborhood of t for which the Hausdorff distance between $\gamma(t)$ and $\gamma(t')$ is less than ε for any t' in the neighborhood. See Figure 1 for an example

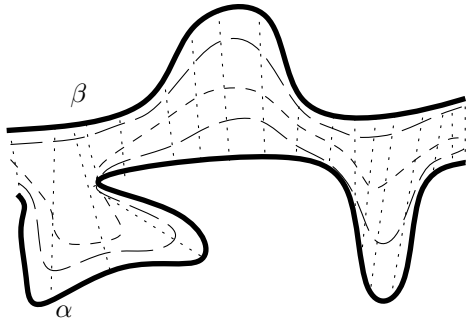


Figure 1: The two polylines α and β and an example of a morphing between them. Dashed lines represent intermediate polylines. Dotted lines show the paths traveled by the points on α and β during the morphing.

of a morphing. We compute a morphing scheme from α to β that consists of two parts. The first part is an explicit mapping between α and β. Given two functions $f : [0, 1] \to \alpha$ and $g : [0, 1] \to \beta$ that parameterize α and β, respectively, we define the mapping as the set of pairs $\{(f(u), g(u)), 0 \leq u \leq 1\}$. We enforce the natural requirement that f and g are monotone along α and β. The second part of the morphing scheme specifies a route connecting every point $f(u)$ to the corresponding point $g(u)$, for $0 \leq u \leq 1$. In this paper, we adopt the policy of moving $f(u)$ to $g(u)$ along the Euclidean shortest path from $f(u)$ to $g(u)$ that avoids α and β. This policy guarantees that all intermediate polylines are simple, since f and g are monotone along α and β and shortest paths do not cross each other (although, two such shortest paths might have a common sub-path). (This property of intermediate polylines being simple does not hold for the Fréchet metric [AG95, Fré06], which corresponds to linking $f(u)$ to $g(u)$ with a straight segment, instead of using shortest paths avoiding α and β, and then optimizing over parameterizations f and g; one can readily construct examples for which $\gamma(t)$ self-intersects for some t.) The resulting morphing scheme is straightforward: we move each point $f(u)$ along its designated route at a constant speed proportional to the length of the route. This morphing is guaranteed to be connected and continuous. Note that once we specify the mapping functions f and g, the morphing scheme is completely determined.

Given these requirements on the morphing scheme, it is possible to generate an uncountable number of different morphing schemes. Clearly, some schemes are better than others. In an effort to formalize a quantitative notion of the quality of a morphing scheme, we define the *width* $W(f, g)$ of a morphing scheme specified by the functions f and g to be the maximum over all values of u of the length of the route connecting the point $f(u)$ on α to the point $g(u)$ on β. We seek morphing schemes that minimize the width over all possible morphing schemes, i.e., a morphing scheme whose width is $W(\alpha, \beta)$, the geodesic width of α and β.

In this part of the paper, we establish three main results. We first show that if n is the total number of vertices of α and β, we can compute $W(\alpha, \beta)$ and a morphing scheme with that width in $O(n^2 \log^2 n)$ time using $O(n^2)$ space. To develop this algorithm, we study the *shortest path diagram*, a combinatorial structure that encodes all of the shortest paths connecting points on α to points on β. We show that the shortest path diagram has size

$O(n^2)$ and that we can construct it in $O(n^2 \log n)$ time using $O(n \log^3 n)$ working space.[2] We also show that, for any parameter $r > 0$, we can represent using $O(n^2)$ space all shortest paths whose length is at most r. Our algorithm uses this fact in combination with parametric search [Meg83]. In this respect, our algorithm is similar to the technique used by Alt and Godau [AG95] to compute the Fréchet distance between two polylines. However, we need to establish several properties of the shortest path diagram in order to obtain our results.

We also consider the problem of computing morphing schemes that treat α and β as being "rigid." Consider the case in which α and β correspond to physical objects. It may be desirable to control the extent by which a portion of α or β is stretched or compressed by the morphing scheme. Given a constant $\kappa > 1$, we restrict our attention to morphing schemes that stretch or compress any portion of α or β by at most a factor of κ; we show that we can compute in $O(n^2 \log^2 n)$ time using $O(n^2)$ space a morphing of minimum width among all such morphings.

Finally, we also describe an algorithm that computes in $O(n \log n)$ time a morphing scheme whose width is at most $2W(\alpha, \beta)$.

1.2 Sweeping Polygons

In the second part of this paper, we focus on computing a motion plan inside a simple polygon for multiple mobile guards whose goal is to "see" targets inside the polygon, or to verify that no target is present in the polygon. Nothing is known about the location of the targets or their motion abilities, except that their motion must be continuous. The guards see a target when there is an unobstructed line-of-sight between one of the guards and the target. We may impose various limitations on the viewing frustum and the range of the vision sensors of the guards.

Parsons [Par76] and Megiddo et al. [MHG+88] 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 at 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. [CSY95], Suzuki and Yamashita [SY92], Guibas et al. [GLL+97], and LaValle et al. [LLG+97] 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

[2] We use $O(n \log^3 n)$ space if we are allowed to output the elements of the shortest path diagram as we compute them. If we are required to store the entire shortest path diagram, we need $O(n^2)$ space.

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* [Hef96, IK92, Kle91, THL98]. In the framework of Icking and Klein [IK92], 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, Heffernan [Hef96] shows that $O(n)$ time suffices to check whether a sweep by two guards exists between p and q and Icking and Klein construct such a sweep in $O(n \log n + k)$ time, where k is the number of "walk" instructions given to the guards to implement the sweep. Tseng et al. [THL98] 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 work by Suzuki and Yamashita [SY92], Lee, Park, and Chwa [LPC02] and Tan [Tan00] have given techniques to check in $O(n^2)$ time if a chain of two or three guards can sweep a polygon (i.e., if there exists a search schedule for a 1-searcher or a 2-searcher, in their terminology).

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). The relationship between r^* and the link width, $\mathcal{L}(\alpha, \beta)$, defined earlier is as follows: The boundary, ∂P, of P can be partitioned into two polylines, α and β, such that $\mathcal{L}(\alpha, \beta) = r^*$, and there exists no partitioning of ∂P into two polylines having link width strictly less than r^*.

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

An intermediate result we achieve is one that uses $O(n^2)$ time and $O(n^2)$ space to 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 $O(rn^2 \log r)$ time a search schedule of size $O(rn^2)$ for P that uses $r + 4$ guards. We also 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 $O(rn)$. (The link radius of P is the minimum over all points $p \in P$ of the maximum link distance from p to other points of P.)

The primary difficulty in planning motions for more 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", 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.

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 breadth" of P. Surprisingly, we can show that r^* is bounded from above and from below by the link breadth (ignoring additive constants).

2 Computing the Morphing Width Exactly

2.1 Geometric Preliminaries

We assume that $\alpha \cup \beta$ is a closed Jordan path, i.e. that α and β have common endpoints a and b. This assumption is without loss of generality since, if this is not the case, we augment $\alpha \cup \beta$ with two curves, γ_1 and γ_2, of minimum total length, not crossing α or β, that match the endpoints of α with the endpoints of β. Since the total length is minimized, a simple exchange argument shows that the curves γ_1 and γ_2 do not cross each other. Thus, $\alpha \cup \gamma_1 \cup \beta \cup \gamma_2$ forms a closed Jordan path, so we can extend α and β in such a way that they have common endpoints.

Let P be the simple polygon whose boundary, ∂P, is the union of α and β. For two points $p, q \in P$, let $\pi_P(p, q)$ denote the shortest path in the plane between p and q that lies inside P and let $d_P(p, q)$ denote the length of this path. We will drop the subscript when P is clear from the context. For two points $p, q \in P$, it is well-known that $\pi(p, q)$ is a polygonal chain whose vertices (other than p and q) are reflex vertices of P. We say that the *combinatorial structure* of $\pi(p, q)$ is the following sequence: the edge of P on which p lies, the sequence of reflex vertices of P through which $\pi(p, q)$ passes, and the edge of P on which q lies.

We assume, without loss of generality, that both α and β have unit length; otherwise, we can use a simple rescaling in the parameterization. Let $f : [0, 1] \to \alpha$, where $f(0) = a$ and $f(1) = b$ be a continuous, increasing[3] function. Let $g : [0, 1] \to \beta$, where $g(0) = a$ and $g(1) = b$ be a continuous, increasing function. We say that the pair of functions (f, g) induces a *mapping* from α to β that associates the point $f(u) \in \alpha$ with the point $g(u) \in \beta$, where $0 \leq u \leq 1$. We define the *width* $W(f, g)$ of a mapping to be $\max\{d(f(u), g(u)), 0 \leq u \leq 1\}$. The following lemma states a useful property of the total number of combinatorially-distinct paths $\pi(f(u), g(u)), 0 \leq u \leq 1$.

[3]The function f is increasing if for every $0 \leq u_1 < u_2 \leq 1$, $f(u_1)$ is closer to a along α than is $f(u_2)$.

Lemma 2.1 *Let (f, g) be a pair of functions inducing a mapping from α to β. Let $0 \leq u_1 < u_2 < \ldots < u_k \leq 1$ be numbers such that $\pi(f(u_i), g(u_i))$ and $\pi(f(u_j), g(u_j))$ are combinatorially different ($1 \leq i < j \leq k$). Then $k = O(n)$ and we can store a representation of the set of shortest paths $\{\pi(f(u_i), g(u_i)), 1 \leq i \leq k\}$ using $O(n)$ space.*

Proof: Clearly if a vertex v of P appears on $\pi(f(u_j), g(u_j))$ but not on $\pi(f(u_{j+1}), g(u_{j+1}))$, then v does not appear on any path $\pi(f(u_k), g(u_k))$ for $k > j$, since $\pi(f(u_{j+1}), g(u_{j+1}))$ separates P into two regions, and v lies in the interior of only one of them. From this observation, it also follows that we can store the difference between every two consecutive paths $\pi(f(u_j), g(u_j))$ and $\pi(f(u_{j+1}), g(u_{j+1}))$ using only $O(n)$ space. □

Given a pair (f, g), for $0 \leq u, t \leq 1$, let $\delta(u, t)$ be the point on $\pi(f(u), g(u))$ such that $d(f(u), \delta(u, t)) = t \cdot d(f(u), g(u))$. The following lemma shows how to convert (f, g) into a morphing.

Lemma 2.2 *Let (f, g) be a pair of functions inducing a mapping from α to β. For $0 \leq t \leq 1$, let $\gamma(t)$ be the sequence of points $\{\delta(u, t), 0 \leq u \leq 1\}$. The sequence $\gamma(t)$ is connected and simple and the set $\Gamma = \{\gamma(t) \mid 0 \leq t \leq 1\}$ is a continuous morphing scheme from α to β.*

Proof: Clearly for any choice of $\xi > 0$ we can find an $\varepsilon > 0$ such that $|d(f(u), g(u)) - d(f(u+\varepsilon), g(u+\varepsilon))| \leq \xi$, and moreover, the Hausdorff distance[4] between $\pi(f(u), g(u))$ and $\pi(f(u+\varepsilon), g(u+\varepsilon))$ is at most ξ. In particular $||\delta(u, t) - \delta(u+\varepsilon, t)|| \leq \xi$, for every t. Thus, it is an elementary calculus exercise to show that $\{\delta(u, t), 0 \leq u \leq 1\}$ is continuous for any choice of t. Using similar arguments, one can show that for every t, $||\delta(u, t) - \delta(u, t+\varepsilon')|| \leq \xi$, for an appropriate $\varepsilon' = \varepsilon'(u, t) > 0$; thus, the Hausdorff distance between $\gamma(t)$ and $\gamma(t + \varepsilon')$ is at most ξ, implying that Γ is continuous. □

Given these two lemmas, we concentrate in the rest of this part of the paper on computing a mapping (f, g) from α to β and ignore the process of converting such a mapping into a morphing scheme.

2.2 The Shortest Path Diagram

In this section, we define the shortest path diagram of a simple polygon and establish some of its properties. The shortest path diagram represents all of the shortest paths inside P in terms of their combinatorial structures.

Let $o \in \partial P$. We parameterize points $p \in \partial P$ by the clockwise distance from o to p along ∂P, divided by the perimeter of P. Let $\phi : [0, 1) \to \partial P$ denote the bijective function corresponding to this parameterization; thus, $\phi()$ maps every point of ∂P to a point in the interval $[0, 1)$. The *shortest-path diagram* $\mathcal{S}_P$ of P is a decomposition of the unit square into maximally connected regions such that for any points (u_1, v_1) and (u_2, v_2) in the same region, the combinatorial structures of the shortest paths $\pi(\phi(u_1), \phi(v_1))$ and $\pi(\phi(u_2), \phi(v_2))$ are identical. We will use the terms *faces*, *arcs*, and *nodes*, respectively, to denote the regions

[4] The Hausdorff distance between α and β is $\leq \xi$ if and only if for every point x of one of the curves there is a point on the other curve whose Euclidean distance from x is $\leq \xi$ — see, e.g., [AG00] for details.

of $\mathcal{S}_P$, the one-dimensional boundaries between the regions of $\mathcal{S}_P$, and the points where two or more arcs of $\mathcal{S}_P$ meet. Two adjacent faces of $\mathcal{S}_P$ correspond to shortest paths whose combinatorial structures differ by one element. Therefore, every node in $\mathcal{S}_P$ has even degree. We now prove that the size of $\mathcal{S}_P$ is $O(n^2)$. We first characterize the arcs of $\mathcal{S}_P$.

Lemma 2.3 *Let (u, v) be a point on an arc of $\mathcal{S}_P$. Then the shortest path $\pi = \pi(\phi(u), \phi(v))$ satisfies one of the following conditions:*

1. *If π consists of only one edge, then π passes through a vertex of P.*
2. *If π contains two or more edges, then either the first or last edge of π touches two vertices of P.*

Proof: Suppose the conditions of the lemma are not true. Then there exists a square s of suitably small size centered at (u, v) such that for any point $(u', v') \in s$, $\pi(\phi(u), \phi(v))$ and $\pi(\phi(u'), \phi(v'))$ are combinatorially equivalent, which contradicts the fact that (u, v) is a point on an arc of $\mathcal{S}_P$. □

Corollary 2.4 *If (u, v) is a node of $\mathcal{S}_P$, then the first and last edges of $\pi(\phi(u), \phi(v))$ are each incident on two vertices of P.*

A simple consequence of this corollary is that no two nodes of $\mathcal{S}_P$ have identical combinatorial structures. This characterization of the nodes of $\mathcal{S}_P$ let us bound the number of nodes in $\mathcal{S}_P$.

Lemma 2.5 *There are $O(n^2)$ nodes in $\mathcal{S}_P$.*

Proof: For two points $p, q \in \partial P$, we say that the shortest path $\pi(p, q)$ is *special* if both the first and last edges of $\pi(p, q)$ are incident on two vertices of P. By Corollary 2.4, every node in $\mathcal{S}_P$ corresponds to a unique special path. We prove that there are $O(n^2)$ special shortest paths in P, which proves the lemma. Let $\pi(p, q)$ be a special path and let e and f be the edges of $\pi(p, q)$ that are incident on p and q respectively. There are four types of special paths:

1. Both p and q are vertices of P. Clearly, there are $O(n^2)$ such special paths.
2. p is not a vertex of P but q is. In this case, let r be the vertex of P that lies in e and is closer to p. The special path $\pi(r, q)$ is of the previous type, so we can charge $\pi(p, q)$ to $\pi(r, q)$.
3. p is a vertex of P but q is not. We handle this case in a manner similar to the previous case.
4. Neither p nor q is a vertex of P. We can handle this case by combining the arguments for the previous two cases.

Each special path is charged at most four times by this argument, thus proving the lemma. □

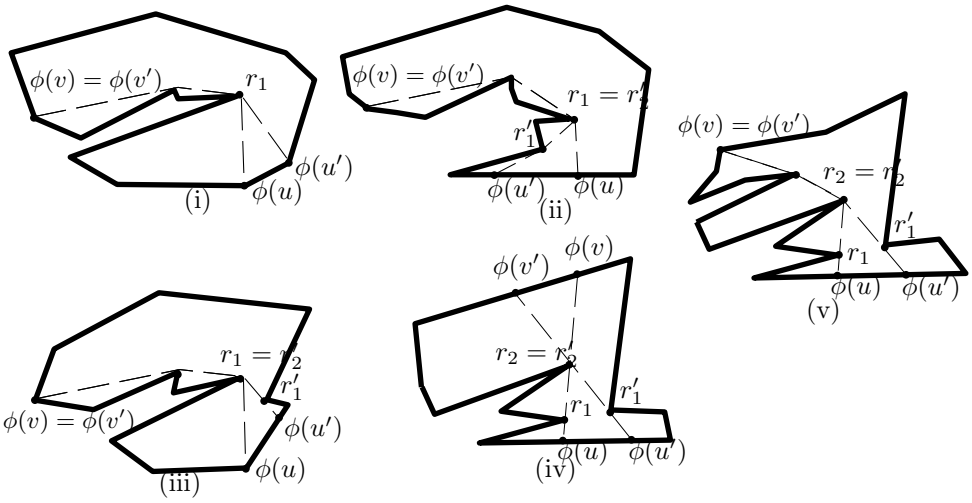


Figure 2: Proof of Lemmas 2.7 and 2.8

Theorem 2.6 *If P is a polygon with n vertices, then its shortest-path diagram has complexity $O(n^2)$.*

In order to describe the algorithm to construct $\mathcal{S}_P$, we prove a property of the nodes of $\mathcal{S}_P$.

Lemma 2.7 *There are four arcs incident on each node of $\mathcal{S}_P$.*

Proof: Let $\pi(\phi(u), \phi(v))$ be the path corresponding to a node (u, v), connecting the points $\phi(u)$ and $\phi(v)$ on ∂P. Clearly both the first and last edges of π are adjacent to two vertices (not necessarily distinct). We need to check two cases:

- $\pi(\phi(u), \phi(v))$ consists of a single segment $\overline{\phi(u)\phi(v)}$ (see for example Fig. 2 (iv)). Assume first that $\overline{\phi(u)\phi(v)}$ is adjacent to two reflex vertices r_1 and r_2 of P, different from its endpoints. By slightly perturbing $\phi(u)$ and $\phi(v)$ along ∂P, we can align $\pi(\phi(u), \phi(v))$ so that among all vertices of P, π touches only r_1, only r_2, neither r_1 nor r_2, or both (and, in this case, π consists of three edges). Clearly these four cases are the only four possible cases, for small enough perturbation of u and v. This is also the only case at which an arc is not horizontal nor vertical.

  It is easily checked that a similar argument holds if either r_1, r_2 or both are also the endpoints of $\pi(\phi(u), \phi(v))$.

- $\pi(\phi(u), \phi(v))$ consists of two or more segments. In this case, we can perturb the endpoints of $\phi(u)$ and $\phi(v)$ independently — see for example Fig. 2(i), (ii), (iii) and (v), where the paths obtained by perturbing only one of them corresponds to the four arcs adjacent to (u, v). Clearly no other arc is adjacent to the node (u, v).

□

The next lemma establishes the relationship between two nodes that are connected by an arc. The intuition behind this lemma is that if two nodes are connected by an arc, then the shortest paths corresponding to these nodes are closely related. In particular, it is true that the first edge on one path is adjacent to the first edge in the other path in the visibility ordering around the first or second vertex of one of the two paths or that an analogous statement holds for the last edges of the two paths.

Lemma 2.8 *Let (u, v) and (u', v') be two nodes of $\mathcal{S}_P$ that are incident on the same arc γ of $\mathcal{S}_P$. Assume that $u \neq u'$. Let $r_1, r_2, \ldots$ be the (ordered) sequence of reflex vertices on $\pi(\phi(u), \phi(v))$ and let $r'_1, r'_2, \ldots$ be the sequence of reflex vertices on $\pi(\phi(u'), \phi(v'))$. Assume that r_1 and r'_1 are the vertices closer to $\phi(u)$ and $\phi(u')$, respectively, in each sequence. Then one of the following conditions holds:*

(i) The paths $\pi(\phi(u), \phi(v))$ and $\pi(\phi(u'), \phi(v'))$ pass through the same number of reflex vertices of P, $r_1 = r'_1$, and the triangle with vertices r_1, $\phi(u)$, and $\phi(u')$ does not contain any vertices of P in its interior, (see Fig. 2 (i))

(ii) The path $\pi(\phi(u'), \phi(v'))$ passes through one more reflex vertex than $\pi(\phi(u), \phi(v))$, $r_1 = r'_2$, and the triangle with vertices r_2, $\phi(u)$, and $\phi(u')$ does not contain any vertices of P in its interior (see Fig. 2 (ii) and Fig. 2 (iii)).

(iii) The paths $\pi(\phi(u), \phi(v))$ and $\pi(\phi(u'), \phi(v'))$ pass through the same number of reflex vertices of P, $r_2 = r'_2$, and the triangle with vertices r_2, $\phi(u)$, and $\phi(u')$ does not contain any vertices of P in its interior, (see Fig. 2 (iv) and Fig. 2 (v)).

We are now ready to describe the algorithm for constructing $\mathcal{S}_P$. There are two main steps.

1. We construct the set S of all of the nodes of $\mathcal{S}_P$. To do so, we compute the shortest paths between all pairs of vertices of P using the techniques of Guibas et al. [GHL$^+$87]. We add the node corresponding to each of these paths to S. If we compute the shortest path for a pair of vertices (p, q), where p is a reflex vertex, then we extend the first edge of $\pi(p, q)$ backward till it intersects ∂P at a point p' by performing a ray-shooting query [CEG$^+$94]. We then add the node corresponding to (p', q) to S. We perform a similar process if q is a reflex vertex. As a by-product of the process of computing all of the shortest paths, we also store for each reflex vertex v of P a sorted list L_v of all of the vertices of P that are visible to v.

2. We use Lemma 2.8 to connect all the nodes in S by computing the arcs of $\mathcal{S}_P$. Let (u, v) be a node computed in the previous step and let r and s be the first and last vertices on $\pi(\phi(u), \phi(v))$, respectively. We assume, without loss of generality, that $\pi(\phi(u), \phi(v))$ is directed from $\phi(u)$ to $\phi(v)$ and that r is to the left of $\pi(\phi(u), \phi(v))$.

   (a) We do a binary search to locate $\phi(u)$ in the list L_r. Let p be the vertex in L_r to the left of $\phi(u)$. Let p' be the point on ∂P obtained by extending the segment rp beyond p (we have already computed p' in the previous step). We connect (u, v) to the node $(\phi^{-1}(p'), v)$.

(b) If r' is the second vertex on $\pi(\phi(u), \phi(v))$ and r' lies on the right of $\pi(\phi(u), \phi(v))$, we locate $\phi(u)$ in the list $L_{r'}$, find the point p' in $L_{r'}$ to the right of $\phi(u)$, and connect (u, v) to the node $(\phi^{-1}(p''), v)$, where p'' is the point on ∂P obtained by extending the segment $r'p'$ beyond p'.

(c) We execute an analogous procedure with $\phi(v)$ and s.

The correctness of the algorithm follows from Corollary 2.4 and Lemma 2.8. We now analyze the running time of the algorithm. We can compute the shortest path tree for each vertex of P in $O(n)$ time per vertex [GHL$^+$87]. We can answer each ray-shooting query made in Step 1 in $O(\log n)$ time [CEG$^+$94]. Thus, we spend a total of $O(n^2 \log n)$ time in Step 1. In Step 2, for each arc we add, we spend $O(\log n)$ time. Therefore, the running time of the algorithm is $O(n^2 \log n)$. The algorithm uses $O(n^2)$ space. If it is enough to output the nodes and arcs of $\mathcal{S}_P$ as we find them, we can reduce the space requirement to $O(n \log^3 n)$ using the technique of Agarwal et al. [AAAS94] to store the visibility graph of a polygon compactly.

Theorem 2.9 *We can construct the shortest path diagram of a polygon with n vertices in $O(n^2 \log n)$ time using $O(n \log^3 n)$ working space.*

2.3 The Main Algorithm

In this section, we present an algorithm for computing a mapping (f, g) between α and β whose width is $W(\alpha, \beta)$, the morphing width of α and β. We will use the shortest path diagram $\mathcal{S}_P$ as the basis of our algorithm in combination with parametric search [Meg83]. To make the description of the algorithm simpler, we slightly modify the definition of $\mathcal{S}_P$. Recall that a and b are the common endpoints of α and β and that α and β have unit length. Let $\phi_1 : [0, 1] \to \partial\alpha$ denote the bijective function such that $\phi_1(u), 0 \leq u \leq 1$ is the point on α whose distance from a along α is u. Define $\phi_2 : [0, 1] \to \partial\beta$ similarly. The *shortest-path diagram* $\mathcal{S}_P$ of $P = \alpha \cup \beta$ is a decomposition of the unit square into maximally connected regions such that for any points (u_1, v_1) and (u_2, v_2) in the same region, the combinatorial structures of the shortest paths $\pi(\phi_1(u_1), \phi_2(v_1))$ and $\pi(\phi_1(u_2), \phi_2(v_2))$ are identical. Note that we can use the algorithm of the previous section (with minor modifications) to construct $\mathcal{S}_P$.

We define a *trajectory* to be an xy-monotone path in $\mathcal{S}_P$ connecting the points $(0, 0)$ and $(1, 1)$. Let T be a trajectory in $\mathcal{S}_P$. We observe that there is a unique mapping (f_T, g_T) corresponding to T; for every point $(x, y) \in T$, this mapping associates the point $\phi_1(x) \in \alpha$ with the point $\phi_2(y) \in \beta$. We abuse notation and let $d(u, v)$ denote the shortest-path distance $d(\phi(u), \phi(v))$. We define $d_T = \max_{(u,v)\in T} d(u, v)$ as the maximum value that $d(u, v)$ assumes over all points $(u, v) \in T$. The following simple lemma is central to our algorithm (refer to the introduction for the definition of the geodesic width $W(\alpha, \beta)$):

Lemma 2.10 *If T is a trajectory in $\mathcal{S}_P$ and (f_T, g_T) is the mapping corresponding to T, then the geodesic width $W(f_T, g_T) = d_T$.*

Thus, it is enough to compute the trajectory T^* that minimizes the value of d_{T^*} over all trajectories. We omit the details of constructing the mapping corresponding to T^*.

It appears difficult to compute T^* directly from $\mathcal{S}_P$. We adopt an approach that is similar to the technique used by Alt and Godau [AG95] for computing the Fréchet distance between two polylines. In a pre-processing step, we compute $\mathcal{S}_P$. We then construct an oracle that determines for a parameter $r > 0$, whether there exists a trajectory T_r with $d(T_r) \leq r$. We combine this oracle with parametric search [Meg83] to compute T^*.

For $r > 0$, we define $\mathcal{S}_P(r)$ to be set of all points (u, v) such that $d(u, v) \leq r$; clearly, $\mathcal{S}_P(r)$ is a refinement of $\mathcal{S}_P$. Our oracle is simple: we first construct $\mathcal{S}_P(r)$ and then check if there is a trajectory that passes only through the points of $\mathcal{S}_P(r)$. To construct $\mathcal{S}_P(r)$, we note that in each face of $\mathcal{S}_P$, the function $d(u, v)$ has constant description complexity. Therefore, the complexity of the portion of $\mathcal{S}_P(r)$ inside a face of $\mathcal{S}_P$ is proportional to the size of that face. Thus, the size of $\mathcal{S}_P(r)$ is $O(n^2)$ and we can compute it from $\mathcal{S}_P$ in $O(n^2)$ time. We use a standard graph search to check if there is a trajectory that passes only through points of $\mathcal{S}_P(r)$.

We briefly describe the generic part, which is essentially identical to the method used in [AG95]. We consider the set $\mathcal{X}$ of all of the locally x-extreme points of $\mathcal{S}_P$. As r changes, the points of $\mathcal{X}$ change, and when $r = r^*$, two of them have the same x-coordinate. Following the parametric search paradigm, it is enough to sort the x-coordinates of the points of $\mathcal{X}$ at the unknown value of r^*, while maintaining an interval $[r_{min}, r_{max}]$ in which r^* lies. During the sorting process, we submit critical values r' to the oracle, to find whether r' is larger or smaller than r^*, and each such query shrinks $[r_{min}, r_{max}]$, until at the end of the process, this interval consists of a single point, which is r^*. Thus we obtain the following result:

Theorem 2.11 *Given two non-intersecting, simple polylines α and β, we can compute a mapping (f, g) between α and β whose width is the morphing width of α and β in $O(n^2 \log^2 n)$ time using $O(n^2)$ space.*

We now show how to extend this result when we want to control the amount of distortion α and β undergo during the morphing. We start by formalizing the notion of distortion. Let (f, g) be a mapping from α to β. For $0 \leq u_1 < u_2 \leq 1$, let $f(u_1, u_2)$ denote the portion of α between $f(u_1)$ and $f(u_2)$ and let $|f(u_1, u_2)|$ denote the length of $f(u_1, u_2)$; define similar notation for g. For $\kappa > 1$, we say that the mapping (f, g) is *κ-stretched* if for every $0 \leq u_1 < u_2 \leq 1$, we have that $(1/\kappa)|f(u_1, u_2)| \leq |g(u_1, u_2)| \leq \kappa|f(u_1, u_2)|$. If T is the trajectory corresponding to a κ-stretched mapping, we say that T is κ-stretched; it is easy to see that the slope at every point on T lies between $1/\kappa$ and κ.

In the rest of this section we present an algorithm to find the κ-stretched trajectory T^* that minimizes the value of d_{T^*} over all κ-stretched trajectories. Once again, we construct an oracle that checks, given a parameter $r > 0$, whether there is a κ-stretched trajectory T such that $d_T \leq r$. We omit the details of how we combine this oracle with parametric search.

Let S be the set of cells of the vertical decomposition of the complement of $\mathcal{S}_P(r)$. Each cell in S has constant description complexity; thus the size $|S|$ of S is $O(n^2)$. To describe the algorithm, we will find it convenient to think of these cells as obstacles. We say that a κ-stretched trajectory T is *legal* if it does not contain any point in the interior of a cell in S, i.e., if it lies entirely inside $\mathcal{S}_P(r)$. The following lemma is central to our argument:

Lemma 2.12 *Given $r \geq 0, \kappa \geq 1$, we can find a legal κ-stretched trajectory in $\mathcal{S}_P(r)$ or determine that no such trajectory exists in $O(n^2 \log n)$ time.*

Proof: Let p be a point in $\mathcal{S}_P(r)$. We define a *flashlight* F_p centered at p to be the wedge with p as origin and whose two edges have slope κ and $1/\kappa$, respectively. A point q is *illuminated by* F_p if q is inside F_p and the segment pq does not intersect the interior of any cells in S. A point q is *illuminated* if it is illuminated either by the flashlight $F_{(0,0)}$ or by a flashlight F_p such that p is also illuminated. It is clear that the point $(1,1)$ is illuminated if and only if $\mathcal{S}_P(r)$ contains a legal κ-stretched trajectory. Therefore, we settle the question of whether $\mathcal{S}_P(r)$ contains a κ-stretched trajectory or not by computing the set of all illuminated points and checking if the point $(1,1)$ is illuminated.

We compute the set of illuminated points by performing a sweep starting at $(0,0)$. We use a sweep line ℓ making an angle of $3\pi/4$ with the x-axis and moving to the right. We initially place a flashlight at $(0,0)$ and add more flashlights as the sweep proceeds. At each instant, we maintain the invariant that we have computed the set of illuminated points to the left of ℓ. We also maintain the intervals of ℓ that are illuminated and the intervals in which ℓ intersects the cells of S. The event points of the sweep are the nodes of the cells of S, the intersections of the edges of the flashlights with the arcs of the cells of S, and intersections between the flashlights themselves. The last two types of events are interesting.

1. The edge of a flashlight F_p intersects the boundary of a cell $c \in S$. We trace the illuminated portion of ∂c until we reach a point r where this illuminated portion ends. If the line through p and r is tangent to c and does not intersect the interior of any other cell in S adjacent to r, we add a flashlight centered at r.

2. The intervals of ℓ illuminated by two flashlights intersect. It is easy to see that the union of these two flashlights is connected to the right of ℓ. Therefore, we merge these two flashlights into one.

There are at most $2|S| + 1$ flashlights created during this sweep, since we place at most two flashlights on the boundary of each cell in S. Every time we merge two flashlights, we decrease the number of flashlights by one. Therefore, there are $O(|S|)$ events during the sweep, each of which we can process in $O(\log n)$ time using standard data structures. Since $|S| = O(n^2)$, we have proven the lemma.

It is easy to see that if there is a legal κ-stretched trajectory, then there is a polygonal trajectory whose vertices are the origins of the flashlights placed by the algorithm. Therefore, the algorithm will detect that $(1,1)$ is illuminated. Since the converse is also true, we have established the correctness of the algorithm. □

We obtain the following theorem once we use this lemma in combination with parametric search:

Theorem 2.13 *Given two non-intersecting, simple polylines α and β and a parameter $\kappa > 1$, we can compute a κ-stretched mapping (f, g) between α and β with minimum width in $O(n^2 \log^2 n)$ time using $O(n^2)$ space.*

3 Approximating the Morphing Width

In this section we describe an approximation algorithm for computing the morphing width $W(\alpha, \beta)$ of α and β. As before, a and b are the the two common endpoints of α and β and P is the polygon whose boundary formed by the union on α and β. Let $\sigma = \pi_P(a, b)$ denote the shortest path between a and b inside P. Let P_1 and P_2 be the two polygons created from P by adding σ; see Figure 3(i). The boundary ∂P_1 may be degenerate in the sense that non-consecutive vertices of P_1 have identical coordinates. This degeneracy occurs whenever σ has two consecutive vertices such that one vertex lies in ∂P_1 and the other vertex lies in ∂P_2. A similar remark holds for ∂P_2. We first prove a structural lemma that states that we can approximate $W(\alpha, \beta)$ within a factor of two. Then we show how to compute such an approximation and a morphing scheme with that width.

3.1 A Structural Lemma

The intuition behind our 2-approximation for $W(\alpha, \beta)$ is that the width of the morphing scheme obtained by first transforming α into σ and then transforming σ into β is at most $2W(\alpha, \beta)$. We now provide a formal basis for this intuition. For a point $x \in P$, let $d_P(\sigma, x)$ denote the minimum length of a shortest path between x and a point of σ, and let $w = \max_{x \in \partial P} d_P(\sigma, x)$.

Clearly, any morphing scheme for P has width at least w, since for all $y \in \beta, d(q, y) \geq d(\sigma, q)$ and for all $x \in \alpha, d(x, r) \geq d(\sigma, r)$. We claim that the morphing width of both P_1 and P_2 is at most w, which implies the result. We prove the claim for P_1. A similar argument holds for P_2.

We parameterize α and σ monotonically such that $\alpha = \cup_{s=0}^{1} \alpha(s)$ and $\sigma = \cup_{t=0}^{1} \sigma(t)$. Let $u(s, t)$ denote the length of the shortest path connecting the points $\alpha(s)$ and $\sigma(t)$ inside P_1. By construction, for any $0 \leq a_0 \leq 1$, there exits a b_0 such that $u(a_0, b_0) \leq w$. Furthermore, we claim that if there exists $b_1 \geq b_0$ such that $u(a_0, b_1) \leq w$ then $u(a_0, v) \leq w$, for all $b_0 \leq v \leq b_1$.

Indeed, assume that this claim is false, and let a_0, b_0, and b_1 be such that $b_0 \leq b_1$, $b_1 - b_0$ is minimized, $u(a_0, b_0) = u(a_0, b_1) = w$, and $u(a_0, v) > w$ for all $b_0 < v < b_1$. Let $p = \alpha(a_0), q = \sigma(b_0), r = \sigma(b_1)$, $\gamma_{pq} = \pi_{P_1}(p, q)$, $\gamma_{qr} = \sigma(q, r)$, and $\gamma_{rp} = \pi_{P_1}(r, p)$. Let P' be the polygon bounded by $\gamma = \gamma_{pq} \cup \gamma_{qr} \cup \gamma_{rp}$. We assume without loss of generality that γ is oriented such that the interior of P' lies to the left of γ. See Figure 3(ii). We assume that γ_{pq} and γ_{rp} are disjoint in their interior. Otherwise, the following argument can be applied to their portions that are disjoint.

Note that γ_{pq}, γ_{qr} and γ_{rp} are all portions of shortest paths, and can not be shortened inside P', since $P' \subseteq P$. In particular, except for p, q, r all other vertices of P' have an angle at least π. Following the same reasoning, the angles of P' at q and r must also be at least $\pi/2$ (otherwise, γ_{pq} or γ_{rp} can be shortcut). The angle μ of P' at p is nonzero. Let k be the number of vertices of P'. The arguments above imply that the sum of the angles of P' is at least $(k-3)\pi + 2(\pi/2) + \mu > (k-2)\pi$, which is impossible, since the sum of the angles of a polygon with k vertices is equal to $(k-2)\pi$.

It is now straightforward to prove, that the set

$$U = \left\{ (s,t) \;\middle|\; 0 \le s,t \le 1, u(s,t) \le w \right\}$$

is connected and contains the points $(0,0)$ and $(1,1)$. In particular, U contains an xy-monotone trajectory that connects $(0,0)$ to $(1,1)$. The xy-monotonicity of the trajectory follows from the structure of the Voronoi diagram induced by σ inside P. (See the next section for details.) This trajectory corresponds to a morphing scheme for α and σ with width at most w. This argument proves the following theorem:

Theorem 3.1 *Let $q \in \alpha$ and $r \in \beta$ be the two points that maximize $d_P(\sigma, q)$ and $d_P(\sigma, r)$, respectively. Let $w = \max(d_P(\sigma, q), d_P(\sigma, r))$; then $w \le W(\alpha, \beta) \le 2w$.*

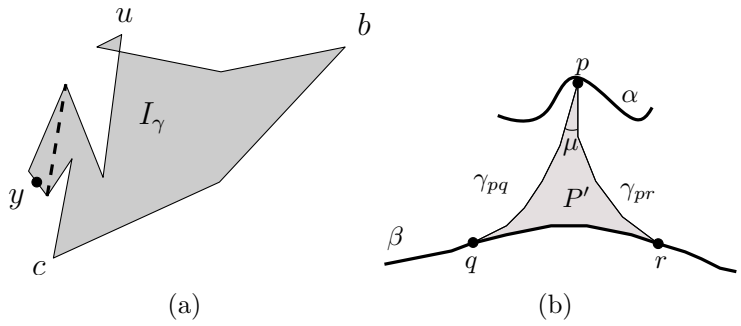


Figure 3: Illustration of the proof of Theorem 3.1: (a) The shortest path σ. (b) The polygon P'. If the angles at q and r are smaller than $\pi/2$ (as in the figure), we can shorten γ_{pq}, γ_{pr} to realize a shorter path between p and a point inside $\sigma(q,r)$.

3.2 Computing a Morphing Scheme with Width $\le 2W(\alpha, \beta)$

In this section, we describe an algorithm that uses Theorem 3.1 to compute a morphing scheme with width at most $2W(\alpha, \beta)$. We only sketch the algorithm, since the details are straightforward but somewhat tedious. We parameterize α and β as before. For each point $x \in \alpha$, we find the shortest path $\pi(x, \sigma)$ and the point $x' \in \sigma$ that realizes this shortest path. Using this information, we construct a morphing scheme for α and σ. We perform a similar procedure for each point on β to compute a morphing scheme for β and σ. Finally, we merge these two morphing schemes to obtain a morphing scheme for α and β. The key step in this algorithm is finding for each point in α and β the closest point on σ. We spend the rest of the section describing our procedure for doing so in $O(n \log n)$ time. Since we compute the curve σ in linear time using the algorithm of Guibas et al. [GHL$^+$87], we obtain the following theorem:

Theorem 3.2 *We compute in $O(n \log n)$ time a morphing scheme (f, g) for α and β such that the width $W(f, g)$ satisfies $W(\alpha, \beta) \le W(f, g) \le 2W(\alpha, \beta)$.*

We now turn our attention to the problem of computing for each point on α the point on σ that is closest to it. We adopt an identical procedure for β. For a point $x \in P_1$, let $v(x)$ be the point of σ such that the length of the Euclidean shortest path $\pi_{P_1}(x, v(x))$ is the smallest over all the paths from x to points on σ; $v(x)$ is either a vertex of σ or a point that lies in the interior of an edge of σ. We now define the *geodesic Voronoi diagram of σ inside P_1*, denoted by $\mathcal{V}_{P_1}(\sigma)$, as the decomposition of P_1 into maximal cells such that for any two points x and y in the same cell of $\mathcal{V}_{P_1}(\sigma)$, the shortest paths $\pi_{P_1}(x, v(x))$ and $\pi_{P_1}(y, v(y))$ are combinatorially identical. It is clear that once we compute $\mathcal{V}_{P_1}(\sigma)$, we compute for each point on α the point on σ that is closest to it. We are not aware of any near-linear time algorithm for computing such a Voronoi diagram in the general case (when σ is an arbitrary polyline and not a shortest-path). In the rest of the section, we drop the subscript from $\mathcal{V}_{P_1}(\sigma)$.

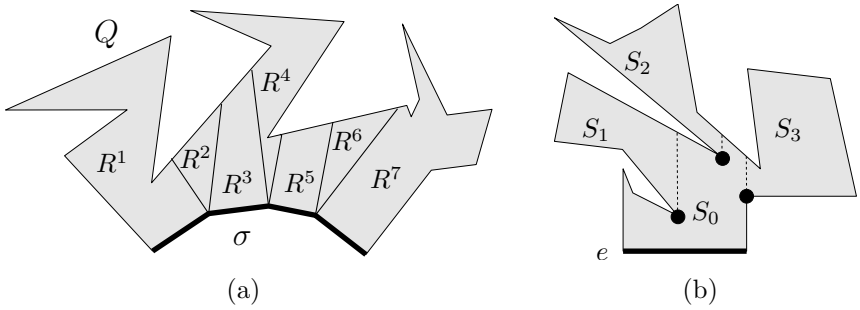


Figure 4: Illustration of the proof of Lemma 3.3: (a) Decomposing a sub-polygon Q into cells of the Voronoi diagram. (b) Decomposing the Voronoi cell corresponding to the edge e by performing vertical sweeping. The resulting sub-polygons are S_0, S_1, S_2, and S_3. Each polygon except S_0 has an associated gateway; all shortest paths from inside such a polygon to e pass through the gateway.

Let $\mathcal{Q}$ be the set of polygons that are the closures of the maximal connected components of the interior of P_1. Clearly, the overall combinatorial complexity of the polygons in $\mathcal{Q}$ is bounded by $O(n)$. We compute the portion of $\mathcal{V}(\sigma)$ inside each polygon $Q \in \mathcal{Q}$ separately. Let σ' denote the portion of σ inside Q. The portion of $\mathcal{V}(\sigma)$ inside Q is identical to the Voronoi diagram $\mathcal{V}_Q(\sigma')$ of σ' inside Q. However, since σ' cannot be shortened inside Q and since ∂Q is not degenerate, we conclude that σ' is concave.

The Voronoi diagram of σ' inside Q is induced either by the interior of an edge of σ' or by a vertex of σ'. The bisector of the interior of an edge of σ' and its endpoint is simply a ray perpendicular to the edge emanating from the vertex directed into the interior of Q. Each vertex of σ' induces two such rays and the region between them is the cell of $\mathcal{V}_Q(\sigma')$ corresponding to the vertex. Since σ' is concave, these rays do not intersect. In particular, these rays decompose Q into a set of polygons $R^1, R^2, \ldots, R^u$ such that each such polygon is the cell in $\mathcal{V}_Q(\sigma')$ corresponding to either a vertex of σ' or an edge of σ'.

We compute the decomposition of Q into the polygons $R^1, R^2, \ldots, R^u$ by preprocessing Q for ray shooting and performing $u - 1$ ray shooting queries in Q. If we use the algorithm of Guibas et al. [GHL$^+$87] to perform these queries, this procedure takes a total of

$O(\sum_{Q \in \mathcal{Q}} |Q| \log |Q|) = O(n \log n)$ time, where $|Q|$ denotes the number of vertices of Q.

We are now left with the task of computing the portion of $\mathcal{V}_Q(\sigma')$ inside each R^j. If R^j corresponds to a vertex of σ', the desired decomposition of R^j is given simply by the shortest path map (rooted at that vertex), which is computed in $O(|R^j|)$ time [GHL$^+$87]. If R^j corresponds to an edge e of σ', we observe that e is on the boundary of R^j and we would like to compute the Voronoi diagram $\mathcal{V}_{R^j}(e)$ of e inside R^j. Rotate R^j so that e is parallel to the x-axis (note that the two edges adjacent to e are vertical). For a point $x \in R^j$, the shortest-path to e ends in a vertical segment connected to e. From each vertex of R^j that is vertically visible from e (we assume that all the vertices that see e are above it), if we shoot a vertical ray upwards until the ray intersects ∂R^j, we decompose R^j into a set $\mathcal{S}$ of polygons. One of the polygons $S \in \mathcal{S}$ has e on its boundary; for every point $x \in S$, $\pi_S(x, e)$ is a vertical segment. Every other polygon $S' \in \mathcal{S}$ has a vertex $p(S')$ that is vertically visible from e. For every point $x \in S'$, $\pi_{S'}(x, e)$ passes through $p(S')$. Thus, we now compute the Voronoi diagram of e inside R^j by computing the shortest path map of $p(S')$ inside each polygon $S' \in \mathcal{S}$.

We compute the set of vertices $\{p(S'), S' \in \mathcal{S}\}$ in $O(|R^j| \log |R^j|)$ time (in fact, we do so in linear time by computing the vertical decomposition of R^j [Cha91]). Inside each sub-polygon $S' \in \mathcal{S}$, we compute the shortest path map of $p(S')$ in $O(|S'|)$ time. Thus, the overall running time of the algorithm is $O(n \log n)$.

Lemma 3.3 *If P is a polygon with n vertices and σ is a concave chain forming a portion of ∂P, we compute the Voronoi diagram induced by σ inside P in $O(n \log n)$ time.*

4 Sweeping Polygons: An Exact Algorithm

4.1 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 on ∂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 configuration of $\mathcal{G}$ defines a piecewise-linear path (the *configuration chain*) 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 chain defined by $\Gamma(t)$; $A_P(0) = 0$. The portion of P that lies to the right of the configuration chain is said to be *clear*; the portion of P to the left of the configuration chain is said to be *contaminated*.

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 5 for an example of such a sweep. We will see (Figure 10) later that there are n-vertex polygons that are not $o(n)$-searchable.

Our algorithms will compute a search schedule for P; while we do not explicitly specify the γ_i's, they can be readily computed from the output of the algorithms.

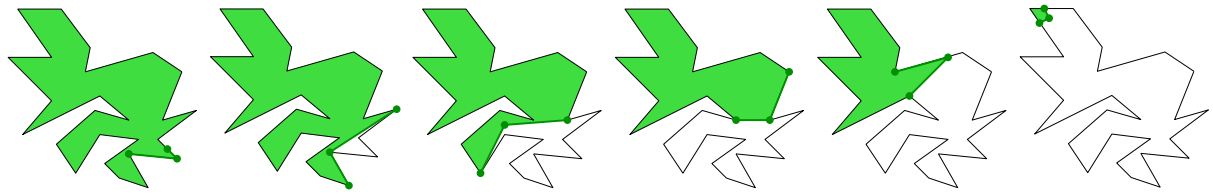


Figure 5: A search schedule with three guards. The unswept ("contaminated") region is shown shaded.

We assume without loss of generality that P has unit perimeter (the length of ∂P is 1) and 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. Thus, at the beginning of the sweep, all of the polygon is contaminated and at the end of the sweep all of the polygon is clear. The following lemma, whose proof follows easily from the definitions, characterizes when a motion strategy is a search schedule:

Lemma 4.1 *Given a motion strategy $(\gamma, \mathcal{G})$, let δ_1 (resp., δ_r) denote the total distance that G_1 (resp., G_r) travels in the counterclockwise (resp., clockwise) direction during γ. If $|\delta_1 + \delta_r| = 1$, then $(\gamma, \mathcal{G})$ is a search schedule for P.*

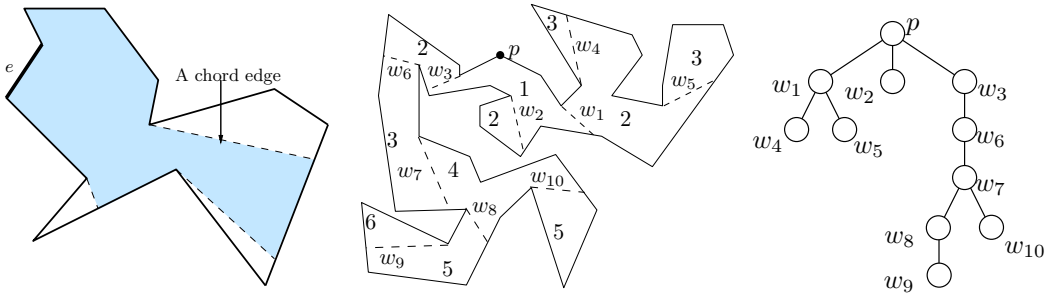
Using this lemma, it is easy to show that in any search schedule, each point in P is swept over an odd number of times.

In all of our algorithms, we construct search schedules in which each configuration of the guards corresponds to a "minimum-link" path between the first and last guards.

We now review some standard definitions related to such paths. 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. For a point $p \in P$, the *visibility polygon* V_p is the set of all points in P that see p. Similarly, for a segment $e \subset P$, the *weak visibility polygon* V_e is the set of all points in P that see some point in e. See Figure 6(a) for an example. An edge of V_e is either (i) a portion of an edge of P or (ii) a segment with one endpoint at a reflex vertex of P and the other endpoint on an edge of P. In the second case, we call the edge a *chord* of V_e. See Figure 6(a). A chord s divides P into two or more sub-polygons; we use $P[s; e]$ to denote the sub-polygons not

containing e. 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.

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.



(a) The weak visibility polygon of an edge e.

(b) The window partition for point p.

(c) The window tree for point p.

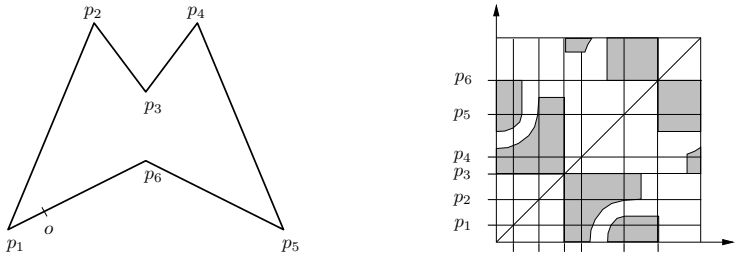
Figure 6: A weak visibility polygon, window partition, and window tree.

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 chord that separates two regions of $\mathcal{W}_p$; we call such a chord a *window* of $\mathcal{W}_p$. See Figure 6(b) for an example. 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 call x the *supporting vertex* of the window and we call y the *far endpoint* of the window. 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 of its windows. The planar dual of $\mathcal{W}_p$ is the *window tree*, $\mathcal{T}_p$. Suri [Sur90] 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 in which the source is a line segment, instead of a point.

We can use the window partition $\mathcal{W}_p$ to compute a minimum-link path from p to any other point in P. In general, minimum-link paths are not unique. The *canonical minimum-link path* $\pi_L(p, q)$ between $p \in P$ and $q \in P$ 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, which is said to *support* the link. (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 are locally supporting the link on opposite sides of the link.

4.2 The Link Diagram

We now define the link diagram of P, a structure that is central to our algorithm for computing r^*, and is analogous to the shortest path diagram defined in Section 2.2. 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 the (unit-length) boundary ∂P. Let $\phi : [0, 1) \to \partial P$ denote the bijective function corresponding to this parameterization; thus, ϕ maps each point in ∂P to a point in the interval $[0, 1)$. For any point (x, y) in the unit square, we abuse notation slightly by letting $d_L(x, y)$ denote the link distance between the points $\phi(x) \in \partial P$ and $\phi(y) \in \partial P$. The *link diagram* $\mathcal{L}_P$ is defined analogously to the shortest-path diagram to be the decomposition of the unit square into maximally connected regions such that the combinatorial type is the same for all paths corresponding to the points within a region. See Figure 7 for an example of $\mathcal{L}_P$. A face of $\mathcal{L}_P$ is a maximally-connected region for which the function d_L assumes the same value; an arc of $\mathcal{L}_P$ separates two different faces of $\mathcal{L}_P$ (the values of d_L in these two faces differ by 1); and a node of $\mathcal{L}_P$ is a point on the boundary of four or more[5] faces of $\mathcal{L}_P$ or a point adjacent to two different arcs that separate the same pair of faces.



(a) (b)

Figure 7: (a) A polygon P and (b) its link diagram. Shaded areas correspond to pairs of points on ∂P with link distance two.

4.3 The Complexity of the Link Diagram

We obtain a tight bound on the worst-case complexity of the link diagram, $\mathcal{L}_P$:

Theorem 4.2 *The link diagram $\mathcal{L}_P$ of a polygon P with n vertices has size $\Theta(n^3)$ in the worst case.*

Proof: Consider any vertical line $\ell(t)$, for $0 \leq t \leq 1$, and the corresponding slice of the $\mathcal{L}_P$. The boundary, ∂P, of P is decomposed into $O(n)$ pieces by the vertices of P and the far endpoints of the windows of $\mathcal{W}_{\phi(t)}$. This decomposition exactly corresponds to the restriction of $\mathcal{L}_P$ to $\ell(t)$. Specifically, $\ell(t)$ crossing an arc of $\mathcal{L}_P$ corresponds exactly to the point $\phi(u)$ coinciding with a vertex of P or with a far endpoint of a window of $\mathcal{W}_{\phi(t)}$. See Figure 8.

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

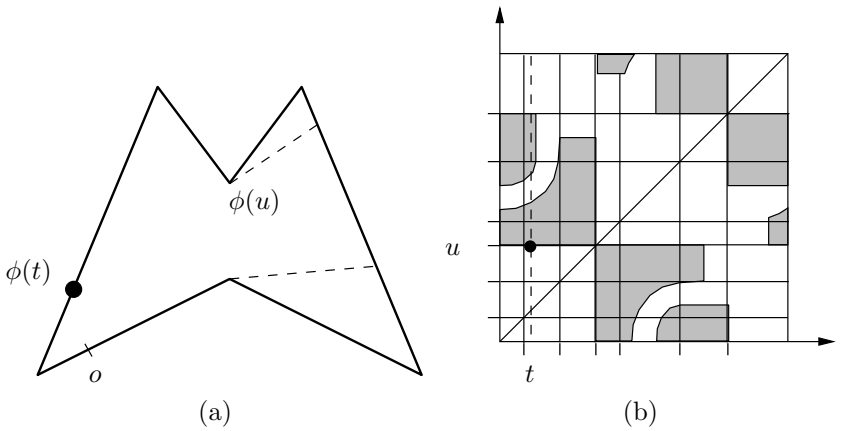


Figure 8: The vertical line $\ell(t)$ intersects $\mathcal{L}_P$ at (t, u) and $\phi(u)$ is the supporting vertex of a window of $\mathcal{W}_t$.

Now consider how the vertical slice of $\mathcal{L}_P$ varies as we vary $t \in [0, 1]$. As t varies, and $\phi(t)$ slides around the boundary of P, the window partition $\mathcal{W}_{\phi(t)}$ changes. In particular, each window λ of $\mathcal{W}_{\phi(t)}$ for which the canonical minimum-link path from $\phi(t)$ to λ has no pinned links will pivot continuously about its supporting vertex. The combinatorial type of $\mathcal{W}_{\phi(t)}$ changes at certain critical values of t, when a window, λ, of $\mathcal{W}_{\phi(t)}$ comes in contact with a vertex, v, of P. At such an event, $\ell(t)$ passes through one or more nodes of $\mathcal{L}_P$, corresponding to the fact that the combinatorial type of paths from $\phi(t)$ to points of ∂P in the sub-window tree associated with λ may have changed.

In fact, Arkin et al. [AMS95] show that the combinatorial type of $\mathcal{W}_{\phi(t)}$ changes at the $O(n^2)$ values of t that correspond to far endpoints of windows in the window partitions $\mathcal{W}_v$ rooted at the n vertices of P. Since each vertical "strip" of $\mathcal{L}_P$ between any two event values of t has $O(n)$ arcs (swept out by the endpoints of windows of $\mathcal{W}_{\phi(t)}$ for values of t within the strip), and there are only $O(n^2)$ event values of t, we get an overall complexity of $O(n^3)$. (Note too that the nodes of $\mathcal{L}_P$ lie on a total of $O(n^2)$ vertical (or horizontal) lines.)

The upper bound of $O(n^3)$ is tight in the worst-case: There are polygons for which the link diagram has size $\Omega(n^3)$. In Figure 9 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 $c_i, 1 \leq i \leq n$ denote the "base" of each tooth in γ_2 and let $d_i, 1 \leq i \leq n$ 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 $1 \leq i, j \leq n$. 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 $j, 1 \leq i \leq n$, there is a window w' in $\mathcal{W}_{p_i}$ such that w' has an endpoint $q \in d_j$. 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 function defining the motion of q (with respect to p) changes. (This function will be a fractional linear function, a *homography*; see [AMS95].) 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, $1 \leq i, j \leq n$, which implies that $\mathcal{L}_P$ has size $\Omega(n^3)$. □

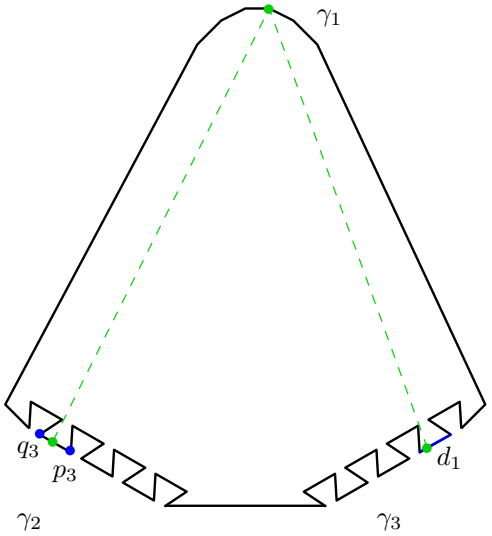


Figure 9: Lower bound construction for the size of $\mathcal{L}_P$.

4.4 Computing the Link Diagram

We now describe an algorithm to construct $\mathcal{L}_P$. The algorithm is very similar to the preprocessing algorithm of Arkin et al. (Section 3.1 of [AMS95]). 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 $[0, 1)$; $u \in L(t)$ if and only if $\phi(u)$ is an endpoint (either a far endpoint or a supporting vertex) of a window in $\mathcal{W}_t$. If $u \in L(t)$, we use $\sigma(t, u)$ to denote the arc of $\mathcal{L}_P$ on which the point (t, u) lies, and we store the combinatorial type of $\sigma(t, u)$ with u in $L(t)$.

1. For each vertex $v \in P$, we compute $\mathcal{W}_v$. This gives $O(n)$ windows per v, for a total of $O(n^2)$ windows overall. We let Q denote the sorted sequence of the vertices of P and these $O(n^2)$ window endpoints, sorted around ∂P. Computing Q requires $O(n^2 \log n)$ time. The points Q induce a partitioning of ∂P into segments ("atomic segments" in [AMS95]). By Lemma 1 of [AMS95], we know that the combinatorial type of $\mathcal{W}_t$ remains constant for points $\phi(t)$ in an atomic segment.

2. For each atomic segment s, we compute the window partition $\mathcal{W}_s$. This takes a total of $O(n^3)$ time, since any one $\mathcal{W}_s$ is computed in linear time. As in step (P4) of the preprocessing algorithm in [AMS95], we keep track, for each window λ of $\mathcal{W}_s$, of the coefficients that specify the homography (fractional linear function) that describes how the far endpoint of λ varies with t, for $\phi(t) \in s$. This can be done easily during the construction of $\mathcal{W}_s$, as observed in [AMS95]. These functions describe the equations of arcs $\sigma(t, u)$ that lie within the vertical strip of $\mathcal{L}_P$ corresponding to atomic segment

s. (In addition to these curved arcs, $\mathcal{L}_P$ has horizontal arcs corresponding to reflex vertices that are supporting vertices of windows λ in $\mathcal{W}_s$.)

In conclusion, we have shown:

Theorem 4.3 *We can construct $\mathcal{L}_P$ in $O(n^3)$ time, using $O(n^2)$ working space.*

4.5 Computing an Optimal Search Schedule

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.

Theorem 4.4 *One can compute r^* by searching $\mathcal{L}_P$, in $O(n^3)$ time. Within the same time bound, one can produce a search schedule of $O(r^*n^3)$ complexity for P using r^* guards.*

Proof: Lemma 4.1 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 4.1 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. 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 $O(r^*n^3)$. □

Remark. In the worst case, r^* may be $\Omega(n)$, since there are n-vertex polygons that are not $o(n)$-searchable. Figure 10 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.

5 Sweeping Polygons: Approximation Algorithms

We have obtained three approximation results: (1) an algorithm that uses $O(n \log n)$ time to compute r^* within an additive error of at most 16, (2) an algorithm that uses $O(n^2)$ time to compute r^* within an additive error of two, and (3) a method for sweeping P that

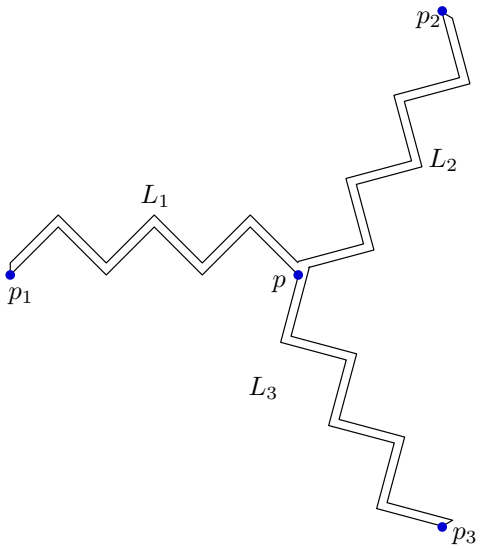


Figure 10: A polygon P such that $r^* = \Omega(n)$.

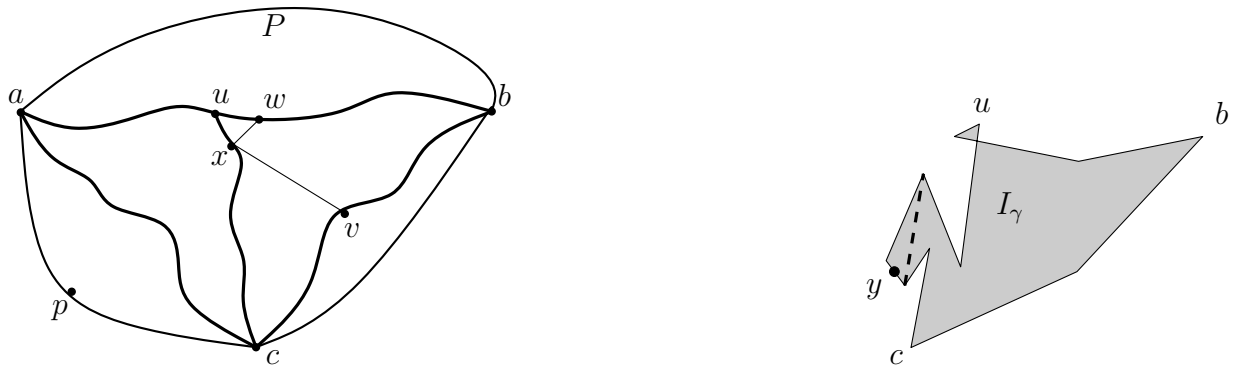
uses at most the link radius of P (which we can compute in $O(n \log n)$ time [DLS92]) plus two guards. Here, we give details of only the first result, (1), which closely parallels the approximation algorithm we already described in the geodesic case; we defer the other two methods to the appendix.

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. (There may be many minimum-link paths that attain the diameter; we fix one of them arbitrarily to be D_P.)

We define a concept that is very similar to that of link width, $\mathcal{L}(\alpha, \beta)$, which we defined in the introduction. In order to distinguish it and hopefully avoid confusion, we refer to this new concept as the "link breadth." In particular, we define the *link breadth of P relative to D_P* to be $\mathcal{L}(P, D_P) = \max_{v \in P} d_L(D_P, v)$. The link breadth of P is then defined to be the minimum, $\min_{D_P} \mathcal{L}(P, D_P)$, taken over all realizations of the diameter. (It turns out that different realizations of D_P can result in different breadths, 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 breadth $\mathcal{L} = \mathcal{L}(P, D_P)$. 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 breadth and the link diameter of P.

Lemma 5.1 *Let $D_P = \pi_L(a, b)$ be a diameter of P, let c be a point that realizes the breadth, $\mathcal{L} = d_L(D_P, c)$, and let u be a point on D_P that is closest to c in link distance. (See Figure 11.) Then, $d_L(a, u) \geq \mathcal{L} - 7$ and $d_L(b, u) \geq \mathcal{L} - 7$.*

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.



(a) definitions (b) shortcut is possible.

Figure 11: Definitions for Lemma 5.1

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 its 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 11(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 link distance):

- $d_L(c,x) - 1 \leq d_L(c,v) \leq d_L(c,x) + 1$.
- $d_L(b,w) - 2 \leq d_L(b,v) \leq d_L(b,w) + 2$.
- $d_L(c,x) \leq \mathcal{L} = d_L(c,u) \leq d_L(c,x) + 1$
- $d_L(c,v) + d_L(b,v) - 1 \leq d_L(b,c)$.

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 $\mathcal{L} - 7 = d_L(c,u) - 7 \leq d_L(a,u)$, and, by symmetry, that $\mathcal{L} - 7 \leq d_L(b,u)$. □

Lemma 5.2 *Let $p \in \partial P(c,a)$ and $q \in \partial P(b,c)$. Then $d_L(p,b) \geq \mathcal{L} - 8$, and $d_L(q,a) \geq \mathcal{L} - 8$.*

Proof: We prove that $d_L(p,b) \geq \mathcal{L} - 8$; the second inequality is shown symmetrically. We may assume that u is chosen to be the last point along $\pi_L(a,b)$ (i.e., the closest to b along the path) among all choices of u that realize the link breadth.

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 $d_L(p,b) \geq \mathcal{L} - 8$ (since Lemma 5.1 says that $\mathcal{L} - 7 \leq d_L(b,u)$).

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. □

Lemma 5.3 *The number of guards needed to sweep a polygon P is at least* $\max(\mathcal{L} - 7, 2)$.

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 5.2, we know that in the first two cases $k \geq \mathcal{L} - 8$. In the third case, the chain of guards must cross $\pi_L(a,b)$, which implies that $k \geq \mathcal{L}$. □

Lemma 5.4 *Let $\sigma = (p_1, \ldots, p_m) \subseteq \partial P$ be a connected 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 5.1, 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 $p_{i+1} q_{i+1}$.) For any point $p \in \partial Q_i$, there exists 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 minimum-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 5.5 $\max(\mathcal{L} - 7, 2) \leq r^* \leq \mathcal{L} + 5$.

Proof: Let P_1, P_2 be the two polygons formed by splitting P along $D_P = \pi_L(a, b)$. By Lemma 5.4, P_1, P_2 can be swept with $\mathcal{L} + 3$ 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 $\mathcal{L} + 5$. The lower bound follows from Lemma 5.3. □

Theorem 5.6 *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 [Ke89a, Ke89b, Sur87]. 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 $d \leq \mathcal{L} \leq d + 4$. Set $k = d + 4$. We know by Theorem 5.5, that P can be swept using $k + 5$ guards and that at least $\max(k - 11, 2)$ guards are needed. □

6 Conclusion

In the time since this paper was submitted, Bespamyatnikh [Bes02] has obtained an improvement to one of our results: He gives a simplified algorithm for computing geodesic width that runs in time $O(n^2)$, using $O(n)$ space, improving our time bound by a factor of $O(\log^2 n)$ and our space bound by a factor of $O(n)$.

Finally, we mention two interesting open directions for future research. First, can we find an appropriate extension of our polygon sweeping results to polygonal domains that have holes? Second, what results can be obtained for the natural generalizations of our problems to three dimensions?

Acknowledgments

The authors wish to thank Pankaj Agarwal, Helmut Alt, Danny Halperin, David Lin, and Micha Sharir for helpful discussions concerning the problems studied in this paper. We also thank the referees for many helpful suggestions that improved the paper.

References

[AAAS94] P. K. Agarwal, N. Alon, B. Aronov, and S. Suri. Can visibility graphs be represented compactly? *Discrete Comput. Geom.*, 12:347–365, 1994.

[AG95] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[AG00] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B. V. North-Holland, Amsterdam, 2000.

[AMS95] 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.

[ASS93] B. Aronov, R. Seidel, and D. Souvaine. On compatible triangulations of simple polygons. *Comput. Geom. Theory Appl.*, 3(1):27–35, 1993.

[Bes02] S. Bespamyatnikh. An optimal morphing between polylines. *Internat. J. Comput. Geom. Appl.*, 12(3):217–228, 2002.

[CEG+94] 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.

[Cha91] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6:485–524, 1991.

[CSL98] D. Cohen-Or, A. Solomovici, and D. Levin. Three-dimensional distance field metamorphosis. *ACM Trans. Graph.*, 17(2):116–141, 1998.

[CSY95] 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.

[DLS92] 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.

[EGH+00] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S.B. Mitchell, and T. M. Murali. Sweeping simple polygons with a chain of guards. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 927–936, 2000.

[EGHM01] A. Efrat, L. J. Guibas, S. Har-Peled, and T. M. Murali. Morphing between polylines. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 680–689, 2001.

[Fré06] M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Mathematico di Palermo*, 22:1–74, 1906.

[GH94] L. J. Guibas and J. Hershberger. Morphing simple polygons. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 267–276, 1994.

[GHL+87] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[GLL+97] 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 Data Struct.*, pages 17–30, 1997.

[GS01] C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers and Graphics*, 25(1):67–75, 2001.

[GSL+99] A. D. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Interactive surface decomposition for polyhedral morphing. *The Visual Computer*, 15(9):453–470, 1999.

[GW97] H. Gupta and R. Wenger. Constructing piecewise linear homeomorphisms of simple polygons. *J. Algorithms*, 22:142–157, 1997.

[Hef96] P. J. Heffernan. An optimal algorithm for the two-guard problem. *Internat. J. Comput. Geom. Appl.*, 6:15–44, 1996.

[HKR93] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:850–863, 1993.

[HS95] J. Hershberger and S. Suri. Morphing binary trees. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, pages 396–404, 1995.

[Hug92] J. F. Hughes. Scheduled fourier volume morphing. In *Proc. SIGGRAPH '92*, pages 43–46, 1992.

[HWK94] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 85–92, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[IK92] Christian Icking and Rolf Klein. The two guards problem. *Internat. J. Comput. Geom. Appl.*, 2(3):257–285, 1992.

[KCP92] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Proc. SIGGRAPH '92*, pages 47–54, 1992.

[Ke89a] Y. Ke. An efficient algorithm for link-distance problems. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 69–78, 1989.

[Ke89b] Y. Ke. *Polygon visibility algorithms for weak visibility and link distance problems.* Ph.D. thesis, Dept. Comput. Sci., Johns Hopkins Univ., Baltimore, MD, 1989.

[Kle91] R. Klein. Moving along a street. In *Proc. Computational Geometry: Methods, Algorithms and Applications*, volume 553 of *Lect. Notes in Comp. Sci.*, pages 123–140. Springer-Verlag, 1991.

[KR91] A. Kaul and J. Rossignac. Solid-interpolating deformations: construction and animation of PIPs. In *Proc. Eurographics '91*, pages 493–505, 1991.

[LLG+97] 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.*, April 1997. To appear.

[LPC02] J. Lee, S. Park, and K. Chwa. Simple algorithms for searching a polygon with flashlights. *Inform. Process. Lett.*, 81(5):265–270, 2002.

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. Assoc. Comput. Mach.*, 30(4):852–865, 1983.

[MHG+88] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. On the complexity of searching a graph. *J. Assoc. Comput. Mach.*, 35:18–44, 1988.

[Par76] 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.

[SG92] T. Sederberg and E. Greenwood. A physically based approach to 2-d shape blending. *Comput. Graph.*, 26:25–34, 1992. Proc. SIGGRAPH '92.

[SG01a] V. Surazhsky and C. Gotsman. Controllable morphing of compatible planar triangulations. *ACM Trans. Graph.*, 20(4):203–231, 2001.

[SG01b] V. Surazhsky and C. Gotsman. Morphing stick figures using optimized compatible triangulations. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, page 40, 2001.

[SGWM93] T. Sederberg, P. Gao, G. Wang, and H. Mu. 2-d shape blending: an intrinsic solution to the vertex path problem. *Comput Graph.*, 27:15–18, 1993. Proc. SIGGRAPH '93.

[SR95] M. Shapira and A. Rappoport. Shape blending using the star-skeleton representation. *IEEE Comput. Graph. Appl.*, 15(2):44–50, March 1995.

[Sur87] S. Suri. *Minimum link paths in polygons and related problems.* PhD thesis, Dept. Comput. Sci., Johns Hopkins Univ., Baltimore, MD, 1987.

[Sur90] S. Suri. On some link distance problems in a simple polygon. *IEEE Trans. Robot. Autom.*, 6:108–113, 1990.

[SY92] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21:863–888, 1992.

[Tan00] X. Tan. Searching a simple polygon by a k-searcher. In *Proc. 11th Annu. Internat. Sympos. Algorithms Comput.*, pages 503–514, 2000.

[THL98] 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.

[TO99] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 335–342, 1999.

A Appendix

We give details of the other two approximation methods ((2) and (3)) that were stated at the beginning of Section 5.

A.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 $O(n^2 r)$ time a search 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 link distance between any point of e_i and any point of e_j; namely, $\mathcal{M}_{ij} = d_L(e_i, e_j)+2$, 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 [Sur90].

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 $\mathcal{M}_{ij} - 2$ and $\mathcal{M}_{ij}$.

Lemma A.1 *Let π and π' be two minimum-link paths, each connecting a point on edge f to a point on edge f'. Let $r = d_L(f, f')$. Then, there exists a sweeping strategy, morphing π into π', using at most $r + 3$ guards. Moreover, in time $O(r)$ we can compute a schedule that uses at most $r + 7$ guards, while issuing $O(r)$ commands.*

Proof: Since the link distance between any point of f and any point of f' is at most $r + 2$, there is a morphing strategy between π and π' using at most $r + 3$ guards. Unfortunately, computing this strategy requires the link diagram of P, which is too expensive to compute within the claimed time bound.

Alternatively, we now sketch an algorithm to compute a strategy that uses at most $r + 7$ guards (4 "spare" guards). Let γ be the closed connected curve comprised of π, π' and the relevant portions of f and f', so that $\pi, \pi' \subseteq \gamma$ (note that γ may have self-intersections). Let I_γ denote the interior of the bounded region delimited by γ, and let $I_1, \ldots, I_k$ be the connected components of the interior of I_γ.

We compute a morphing between $\pi_i = \pi \cap \partial I_i$ and $\pi'_i = \pi' \cap \partial I_i$, with the motion restricted to lie inside I_i, for $i = 1, \ldots, k$. Since π and π' are both minimum-link paths, we know that the number of links in π_i and π'_i is the same, up to at most an additive error of 2; otherwise, replacing one of the subpaths (π_i or π'_i) with the other would result in a net decrease in link length of π or π'. This property also implies that we can compute all of these regions in

total time $O(r)$, since we can simply check the jth segment of π for intersection with $O(1)$ segments of π' (namely, segments $j-2, j-1, j, j+1, j+2$).

We do the morphing region by region, starting with I_1. Let $\pi = (u_1, \ldots, u_k)$ and $\pi' = (u'_1, \ldots, u'_{k'})$ be the vertex sequences defining the paths. We know that $k, k' \leq r+1$. Note that the endpoints of π_i lie either at the points (u_1 and u_k) on edges f and f', or at crossing points, where a link of π crosses a link of π'. (We assume, for simplicity of discussion, there are no degeneracies, where links intersect at a vertex.)

Assume that we want to transform π into π'. Consider a general step of the morphing, in which we want to morph the subpath π_i to the path π'_i, with a chain of $K = 4 + |\pi_i|$ guards, where $|\pi_i|$ denotes the number of vertices of π that lie along π_i. Initially there is a guard at each vertex of π_i (including its endpoints, which may be crossing points of π and π'); the spare guards are placed at the first endpoint of π_i.

We triangulate I_i using only diagonals that join a vertex of π_i to a vertex of π'_i. We know that this can be done, since the minimum-link property implies that (1) there can be no diagonal between two vertices of π_i or two vertices of π'_i, and (2) any endpoint of π_i that is a crossing point is an "ear tip," with an associated diagonal between a vertex of π_i and a vertex of π'_i clipping it off. (In the special case in which π_i consists of a single edge, the diagonal that cuts off one crossing point is incident on the other, and I_i is a quadrilateral; this can be handled separately.) Thus, we obtain a triangulation of I_i whose dual graph is Hamiltonian.

Our morphing strategy considers each triangle, τ, in turn along the dual path in the triangulation. We perform the morphing triangle by triangle, "shifting" the chain of guards across each triangle in succession. Suppose that we have completed the morph up to diagonal $u_j u'_l$, so that the partially morphed chain has one guard at each vertex of π'_i up to and including u'_l, and the remaining guards lie along π_i, from u_j onwards, with all spare guards at u_j. We can assume that if the first vertex of π'_i (and π_i) is a crossing point, then the guard that previously was situated there has been advanced along the chain and is now among the spares. Let τ be the next triangle. In particular, if $\tau = u_j u_{j+1} u'_l$ shares an edge ($u_j u_{j+1}$) with π_i and a vertex (u'_l) with π'_i, then we send all of the guards at u_j to vertex u_{j+1}; we know that they stay visible to u'_l and u_{j+1}. If $\tau = u_j u'_l u'_{l+1}$ shares an edge ($u'_l u'_{l+1}$) with π'_i and a vertex (u_j) with π_i, then we send one of the (spare) guards from u_j to u'_{l+1} (while he maintains visibility with the guards at u_j and u'_l).

The fact that this strategy works, without running out of spare guards, follows again from the minimum-link property: There are at most K vertices along *any* chain formed by the path along π'_i to a vertex u'_l, then the diagonal $u'_l u_j$, then the remaining path, along π_i, from the vertex u_j to the end of π_i. This is easily verified, again, by an exchange argument.

Overall, the morphing strategy uses a number of commands proportional to the number of triangles, which is clearly $O(r)$. □

We construct a graph G on the grid $2n \times 2n$, so that two nodes are adjacent in G if and only if 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 than 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 A.1. Such a minimum-weight path σ in G can be computed in $O(n^2)$ time using Dijkstra's algorithm. We conclude:

Theorem A.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 minimum-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 algorithm of Lemma 5.4, one can compute a morphing strategy between π and π_{mid}, and a morphing strategy between π_{mid} and π'. □

A.2 Link radius + 2 number of guards suffice

We now prove that any polygon P is (R_P+2)-searchable by describing an algorithm that constructs a search schedule for P by using at most R_P+2 guards. We assume without loss of generality that all of the guards are initially placed at some point $p \in \partial P$. We set r, the number of guards in $\mathcal{G}$, to be one more than the height of $\mathcal{T}_p$. Let v be a node in $\mathcal{T}_p$ and let w be the window associated with v. Each child of v is associated with a window that is a chord of V_w. We assume that the left-to-right order of v's children corresponds to the clockwise order of the chords of V_w starting at w. The motion strategy that our algorithm constructs corresponds to a modified pre-order traversal of $\mathcal{T}_p$ where we visit a node v before we visit each of v's children. The following recursive procedure describes our algorithm (initially, we invoke this procedure with the root of $\mathcal{T}_p$ and the set $\mathcal{G}$ of all guards):

VISIT$(v, \mathcal{G}')$: v is a node in $\mathcal{T}_P$ and $\mathcal{G}' = \{G_{d+1}, G_{d+2}, \ldots, G_r\}$, where d is the height of v in $\mathcal{T}_p$. Let $w = ab$ be the window associated with v. Suppose v has m children, i.e., V_w has m chord edges. Let $w_i = a_i b_i$ be the window associated with v_i, the ith child of v. We set $b_0 = a$ and $a_{m+1} = b$. For each $1 \leq i \leq m+1$, we perform the following steps:

1. We move guard G_{d+2} from b_{i-1} to a_i and also move guard G_{d+1} simultaneously on w so that G_{d+1} always sees G_{d+2}. See Figure 12.

2. We station G_{d+1} on w such that it can see a_i and b_i and invoke the procedure VISIT$(v_i, \mathcal{G}' \setminus G_{d+1})$.

This completes the description of the algorithm. It is clear that each configuration assumed by the guards is legal. It is easy to construct a motion strategy $(\gamma, \mathcal{G})$ from the

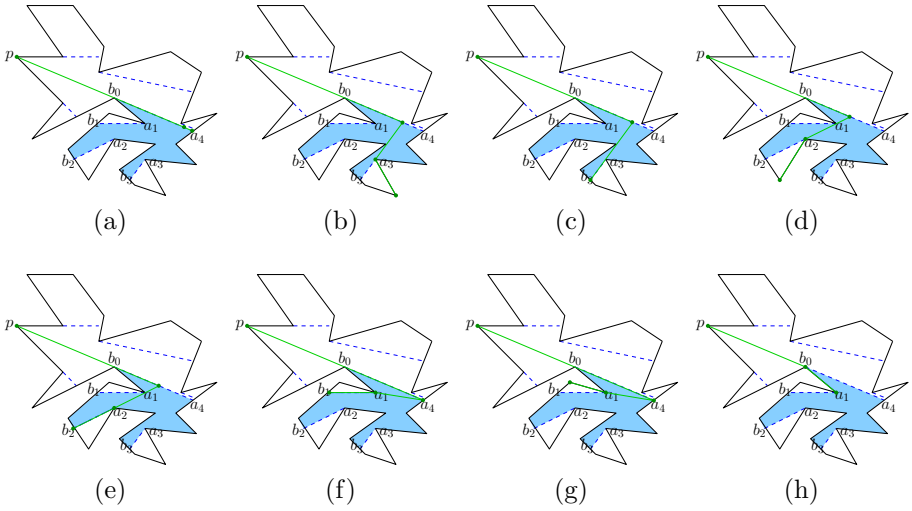


Figure 12: Different stages of the algorithm

algorithm. We now turn our attention to proving that the motion strategy is indeed a search schedule for P and on bounding the number of guards in $\mathcal{G}$.

Lemma A.3 *Suppose v is a node in $\mathcal{T}_p$ and w is the window associated with v. The polygon $P[w;p]$ is clear after the algorithm completes visiting v.*

Proof: Let d be the height of v and let $\mathcal{G}' = \{G_{d+1}, G_{d+2}, \ldots, G_r\}$. Let v' be the parent of v in $\mathcal{T}_p$ and let w' be the window associated with v'. During the invocation of $\textsc{Visit}(v', \mathcal{G}' \cup \{G_d\})$, we visit the children of v' in clockwise order around $V_{w'}$. As a result, after $\textsc{Visit}(v, \mathcal{G}')$ is completed, the configuration of the guards G_d and G_{d+1} does not cross the window w. Since each configuration the guards assume in the algorithm is legal, after $\textsc{Visit}(v, \mathcal{G}')$ is completed, $P[w;p]$ always lies to the same side of the configuration of the guards. This proves the lemma. □

The above lemma has the following corollary:

Corollary A.4 *The motion strategy $(\gamma, \mathcal{G})$ computed by the above algorithm is a search schedule for P.*

It is clear that the maximum number of guards used by the algorithm is one more that the height of $\mathcal{T}_p$. Since the height of $\mathcal{T}_p$ is equal to the maximum link distance from p of any point in P, it is we use at most $D_P + 1$ guards. However, we can improve the number of guards as follows: We compute the link center C of P. If C intersects ∂P, we pick p to be a point in this intersection, thus using $R_P + 1$ guards. Otherwise, we pick p to be any point in ∂P that is seen by a point p' in C. In this case, the height of $\mathcal{T}_p$ is one more than the height of $\mathcal{T}_{p'}$, which implies that we use $R_P + 2$ guards. We have now proved the main result of this section:

Theorem A.5 *For any polygon P with n vertices, we can compute in $O(nR_P)$ time a search schedule for $R_P + 2$ guards that sweep P.*

Remarks: Throughout the motion, guard G_1 is stationed at p. At any stage of the algorithm, the configuration of the guards in $\mathcal{G}$ is a canonical minimum-link path between the positions of the first and last guards.