Finding a Shortest Diagonal of a Simple Polygon in Linear Time

John Hershberger
DEC/SRC
130 Lytton Avenue,
Palo Alto, California 94301

Subhash Suri
Bellcore
445 South Street,
Morristown, New Jersey 07960

August 16, 1993

Abstract

A diagonal of a planar, simple polygon P is an open line segment that connects two non-adjacent vertices and lies in the relative interior of P. We present a linear time algorithm for finding a shortest diagonal (in the L_2 norm) of a simple polygon, improving the previous best result by a factor of $\log n$. Our result provides an interesting contrast to a known $\Omega(n \log n)$ lower bound for finding a closest pair of vertices in a simple polygon—observe that a shortest diagonal is defined by a closest pair of vertices satisfying an additional *visibility constraint*.

1 Introduction

Closest pair problems have been studied in computational geometry for a long time. It is well-known, for instance, that a closest pair among n points in the plane can be found in $O(n \log n)$ time, and that this bound is tight in the worst case in the algebraic-tree model of computation [7]. The lower bound holds even if the points form the vertices of a simple polygon, given in the boundary order [1].

A variant of the closest pair problem that has received some interest recently is the *shortest diagonal* problem. A diagonal of a simple polygon is a line segment joining two non-adjacent vertices that are mutually visible inside the polygon, and a *shortest diagonal* is a diagonal of minimum length. Thus, a shortest diagonal of a polygon is defined by a closest pair among its vertices satisfying the additional constraint that the vertices be visible. Despite the close similarity between the two problems, the lower bound of [1] does not apply to the shortest diagonal problem. The previous best result for computing a shortest diagonal was an $O(n \log n)$ time algorithm, based on a Voronoi diagram [5]. For the special case of a monotone polygon, Zhu [10] was able to get a linear-time algorithm; however, it remained an open question to determine the time complexity of the general problem [6].

In this paper, we settle this question by presenting a linear-time algorithm for computing a shortest diagonal of a simple polygon. A brief sketch of our algorithm is as follows. We partition the polygon into connected, monotone regions each satisfying a property we call *conformity*. The partition is hierarchical, and modeled by a rooted tree, called a *partition tree*. The shortest diagonal within each monotone region is computed using the linear-time algorithm of Zhu [10]. The more complex step involves computing diagonal distances between vertices lying in different regions. Our key lemma shows that the regions containing endpoints of a shortest diagonal cannot be far apart in the partition tree. This allows us to break up the problem into several smaller problems, of total size $O(n)$, each of which requires computing a closest visible vertex pair between two disjoint polygon chains. To solve this last problem, we apply a result from our paper on matrix searching with the shortest path metric [4].

The paper is organized in five sections. Section 2 introduces a few technical preliminaries; Section 3 describes a partition of the polygon, called a conforming partition; Section 4 describes our algorithm; and finally in Section 5 we finish by presenting the details of constructing the conforming partition.

2 Preliminaries

Let P be a simple polygon of n vertices in the plane. We define $\pi(x, y)$ to be the shortest path between the points x and y in P; $d(x, y)$ is the length of the path $\pi(x, y)$. We say that two points x and y are visible in P if the open line segment joining x and y does not intersect the boundary of P. A diagonal of P is a line segment connecting two non-adjacent, visible vertices of P. We want to compute a shortest diagonal of P in linear time.

For technical reasons, it is convenient to dispose of the diagonals defined by vertices that are two edges apart on the boundary of the polygon. We say that two vertices u and v are *2-neighbors* if u and v have a common neighbor on the boundary of the polygon. The following lemma gives a simple linear-time algorithm for computing a shortest diagonal defined by 2-neighbor vertices.

Lemma 2.1 *A shortest diagonal defined by a pair of 2-neighbors can be computed in linear time.*

Proof: We sequentially examine all 2-neighbors, and keep track of the closest visible pair among them. The only non-trivial step of the algorithm is to decide if two candidate vertices are visible. A linear-time preprocessing of the polygon allows us to do this step in constant time. The preprocessing step computes a shortest path tree of P, rooted at an arbitrary vertex x, and records $parent(u)$ for each vertex u in the tree [2, 3]. Now consider an arbitrary 2-neighbor pair (u, v), and let w denote the common neighbor of u and v. Unless $x = w$, the pair u and v is visible if and only if neither $parent(u)$ nor $parent(v)$ lies in the interior of the triangle $\triangle uvw$, which can indeed be checked in constant time. The exceptional case (that is, $x = w$) occurs only once during the entire algorithm, and in that case we spend $O(n)$ time to decide if u and v are visible. This completes the proof. $\square$

As a consequence of Lemma 2.1, we may now limit our attention to only those vertex-pairs that are not 2-neighbors; exclusion of 2-neighbors simplifies some of the technicalities in the forthcoming algorithm. We start the discussion by describing a recursive partition of the polygon into monotone polygons.

3 A conforming partition of the polygon

We partition the polygon P into connected, monotone regions, each satisfying a property we call *conformity*. We describe this partition, denoted $\Pi(P)$, in abstract terms here, deferring the details and an actual construction to Section 5. We use the present and the following section to establish the key properties of our partition and to describe our algorithm for the shortest diagonal problem.

Each region of our partition $\Pi(P)$ is associated with a base edge, where a *base edge* is a horizontal or a vertical chord of P; a chord is a line segment internal to P with both endpoints on the boundary of the polygon. The region associated with a base edge b is denoted $R(b)$, and the base edge b is called the *generator* of $R(b)$. The boundary of $R(b)$ consists of portions of the boundary of P and *windows* in alternating sequence, where a window is a chord of P orthogonal to the generator. Each region $R(b)$ is *monotone* in the direction of its generator b, and does not cross the line supporting b; this implies that all but at most two of the windows of $R(b)$ have a reflex vertex of P at their endpoint nearer b. The two possible exceptions are windows whose supporting lines do not intersect the interior of b and pass on either side of b. Each window, in turn, is the generator for a neighboring region. The recursion stops when the entire polygon is covered by the regions.

The partition $\Pi(P)$ is defined hierarchically, with a natural ancestor-descendant relationship. Each window of a region $R(b)$ is called a *child* of the generator b, and b is the unique *parent* of its windows. We can model this partition by a rooted tree T, called the *partition tree*: each node of T is associated with a unique generator of $\Pi(P)$, with nodes linked by edges directed from parents to children. We denote the generator associated with a node v by $b(v)$, and we denote the region of $b(v)$ by the shorthand notation $R(v)$.

The regions have pairwise disjoint interiors, but each window of one region is the base of a child region. By assigning each window to its parent's region, we can ensure that the regions form a proper partition of the polygon. Finally, we need one more definition: domain of a generator. Each generator is a chord, and it divides P into two subpolygons. The *domain* of a generator b, denoted $P(b)$, is the subpolygon not containing the parent of b; the domain of the root generator is P. Thus, if $u \in T$ is a node, then $P(b(u)) = R(u) \cup \{P(b(v)) \mid v \text{ is a child of } u\}$. Alternatively, $P(b(u)) = R(u) \cup \{R(z) \mid z \text{ is a descendant of } u\}$.

Given an arbitrary point $x \in P$, the *containing* node of x is the unique node $v \in T$ satisfying $x \in R(v)$; by the disjointness of regions, there is exactly one containing node for any point x. The following lemma states a fundamental property of our partition $\Pi(P)$.

Lemma 3.1 *Let x_i and x_j be two arbitrary points in P, and let v_i and v_j, respectively, be their containing nodes. Then the shortest path $\pi(x_i, x_j)$ intersects the region $R(v)$ for every node v on the tree path between v_i and v_j.*

Proof: The lemma depends on the fact that the domain of a generator b contains only the regions associated with the descendants of b. Since b is a chord, any path connecting a point in $P(b)$ to a point outside $P(b)$ necessarily intersects b. By definition, the lowest common ancestor of v_i and v_j is the lowest node whose domain contains both x_i and x_j. The lemma follows. $\square$

The partition described thus far is still quite abstract and general. We need to specialize it further to apply it to the shortest diagonal problem. In particular, we need a *conformity* condition for the regions in our partition, which we now discuss. The conformity condition for a region involves the region's generator, its parent, and its children (windows).

Consider a node $u \in T$, its parent node $p(u)$, and the children of u, say $v_1, v_2, \ldots, v_k$. We call the region $R(u)$ *conforming* if there exists a chord of P parallel to $b(u)$, *passing through a vertex of P,* that separates the base edge $b(p(u))$ from all the windows $b(v_1), b(v_2), \ldots, b(v_k)$. The vertex through which the chord passes in this definition is called a *conformer vertex* of $b(u)$. A partition $\Pi(P)$ is called *conforming* if all of its regions are conforming. An example of a conforming partition is shown in Figure 3.

In the following discussion, we assume that $\Pi(P)$ is a conforming partition.

The linearity of our algorithm depends critically on the following fact: the containing nodes of the vertices defining a shortest diagonal are constant distance apart in the partition tree T. More precisely, let (p, q) be a shortest diagonal, and let v_p and v_q, respectively, be the containing nodes of p and q. Then the lowest common ancestor of v_p and v_q is at most distance two away from v_p and v_q. The proof of this claim requires a somewhat technical lemma, and the following definition of an orthogonal coordinate system.

A triple (L_1, L_2, O) defines an orthogonal coordinate system if L_1 is a horizontal line, L_2 is a vertical line, and L_1 and L_2 intersect at O. This coordinate system has four quadrants, which are numbered in counterclockwise order around O.

Lemma 3.2 *Let p and q be two mutually visible vertices of P that do not form a 2-neighbor pair. Suppose there exists an orthogonal coordinate system (L_1, L_2, O) satisfying the following conditions:*

1. *O is a vertex of P,*
2. *p and q lie in opposite (both even or both odd) quadrants of the coordinate system (L_1, L_2, O), and*
3. *the vertex O is visible from some point on the segment pq.*

Then (p, q) cannot be a shortest diagonal of P.

Proof: Consider the shortest paths $\pi(O, p)$ and $\pi(O, q)$. These paths are contained completely in the triangle $\triangle pOq$.[1] See Figure 1. If an edge of either $\pi(O, p)$ or $\pi(O, q)$ is a diagonal of P, then let (a, b) be that diagonal. In this case, we must have $d(a, b) < d(p, q)$: since $\triangle pOq$ is an obtuse triangle with $\angle pOq$ as its largest angle, $d(p, q)$ is the longest segment contained in $\triangle pOq$. If none of the edges of $\pi(O, p)$ and $\pi(O, q)$ is a diagonal of P, then let p' be the neighbor of p in the path $\pi(O, p)$ and let q' be the neighbor of q in the path $\pi(O, q)$. Since both $\pi(O, p)$ and $\pi(O, q)$ are concave, either p and q' are visible or p' and q are visible. Since p and q are not 2-neighbors, O cannot be adjacent to both p and q. Thus, either (p, q') or (p', q) is a diagonal, and, being contained in $\triangle pOq$, it is shorter than (p, q). In either case, we conclude that (p, q) is not a shortest diagonal of P. $\square$

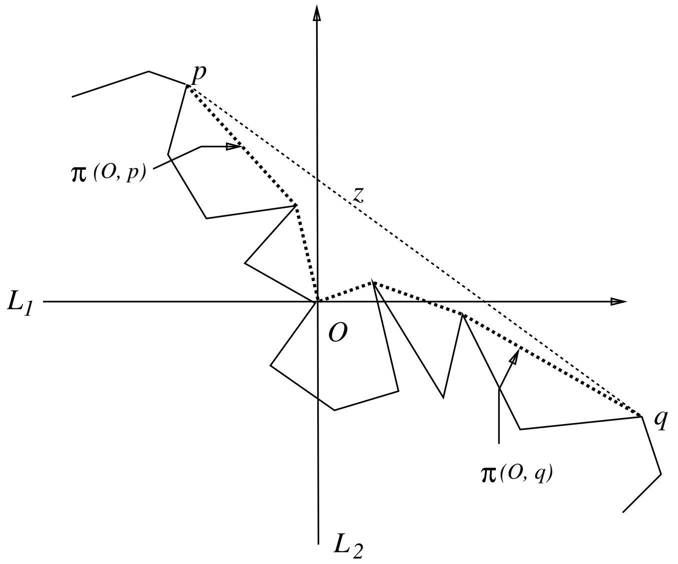

Figure 1: Illustration for the proof of Lemma 3.2.

We introduce one more notation to facilitate our discussion of the tree distance between the containing nodes of two vertices defining a shortest diagonal. Given a node $v \in T$, let $Z_d(v)$ denote the set of vertices in P whose containing nodes are descendants of v at distance at most d from v. In particular, $Z_0(v)$ is the set of vertices in $R(v)$, and $Z_1(v)$ is the set of vertices in $R(v)$ and all the regions associated with children of v.

[1] Think of a rubber-band analogy: initially stretch the rubber band along the path (p, z, O), where z is a point on pq visible from O, and then release the middle of the band while holding the ends fixed. The final rubber-band path coincides with $\pi(O, p)$ and it must remain within the subtriangle $\triangle Opz$. The same holds for $\pi(O, q)$.

Lemma 3.3 *Let (p, q) be a shortest diagonal of P. There exists a node $v \in T$ such that $p, q \in Z_2(v)$.*

Proof: Our proof is by contradiction: we assume that the lemma is false, and show that (p, q) cannot be a shortest diagonal. Let v_p and v_q, respectively, denote the containing nodes of p and q, and let v_{pq} be the lowest common ancestor of v_p and v_q. We prove that the tree distance between v_{pq} and v_p, v_q is at most two, which shows that $p, q \in Z_2(v_{pq})$. Assume, without loss of generality, that the tree distance between v_{pq} and v_p is at least three. Let v'_p and v''_p, respectively, denote the parent and the grandparent of v_p; by assumption, v_{pq} is higher than v''_p. We focus on the generators $b(v_p)$ and $b(v''_p)$.

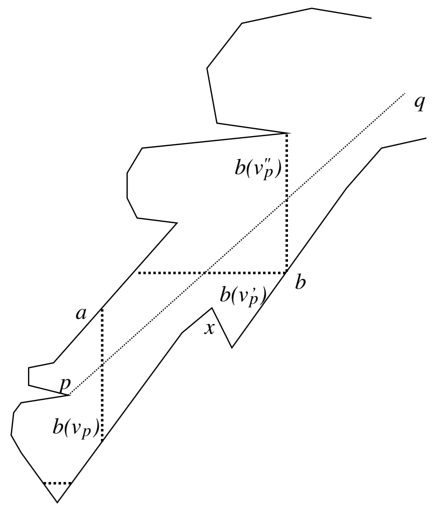

Figure 2: Illustrating the proof of Lemma 3.3. The conformer vertex of $b(v'_p)$ is x.

Since windows alternate between horizontal and vertical along any path in T, the generators $b(v_p)$ and $b(v''_p)$ are parallel. Without loss of generality, assume that (1) $b(v_p)$ and $b(v''_p)$ are both vertical, and (2) $b(v_p)$ is to the left of $b(v''_p)$.

If the endpoint of $b(v'_p)$ closer to $b(v''_p)$ is a vertex of P, then a coordinate system at that vertex satisfies the conditions of Lemma 3.2, and hence (p, q) cannot be a shortest diagonal. Therefore, assume that the endpoint is not a vertex. As noted previously, in this case the line supporting $b(v'_p)$ does not intersect the interior of $b(v''_p)$. Without loss of generality, assume that $b(v'_p)$ is below $b(v''_p)$. Because $b(v_p)$ is below $b(v'_p)$, the projections of $b(v_p)$ and $b(v''_p)$ on a vertical line are disjoint, except possibly for one point. Let a denote the upper endpoint of $b(v_p)$ and let b denote the lower endpoint of $b(v''_p)$. Then the vertex p lies below and left of a, while q lies above and right of b. See Figure 2.

If either a or b is a vertex of P, we can set up a coordinate system at that vertex, satisfying the conditions of Lemma 3.2, which shows that (p, q) cannot be a shortest diagonal. Otherwise, we use the conformer vertex for the parent generator $b(v'_p)$ to produce a desired coordinate system, as follows. By the definition of the conformer vertex, the horizontal chord through it separates the window $b(v_p)$ from $b(v_p'')$. Because a is not a vertex, $b(v_p')$ lies to the right of $b(v_p)$. The base $b(v_p')$ lies to the left of $b(v_p'')$, along with all of $R(v_p'')$. It follows that the x-coordinate of the conformer vertex also lies between those of $b(v_p)$ and $b(v_p'')$. We can set up a coordinate system (L_1, L_2, O), with origin at the conformer vertex, satisfying the conditions of Lemma 3.2, which contradicts our hypothesis that (p, q) is a shortest diagonal. Thus the distance between the lowest common ancestor v_{pq} and v_p, v_q must be at most two. This completes the proof. □

We are now ready to describe our algorithm for computing a shortest diagonal of the polygon.

4 An algorithm for finding the shortest diagonal

Assume that $\Pi(P)$ is a conforming partition of P and that T is the dual tree of $\Pi(P)$. Each region of $\Pi(v)$ is a monotone polygon, and its shortest diagonal can be computed in time proportional to its size using a linear-time algorithm of Zhu [10].

Lemma 4.1 ([10]) *Given a monotone polygon P on n vertices, a shortest diagonal of P can be computed in $O(n)$ time.*

Remark: A region in our partition also has artificial vertices, the endpoints of windows, which should be excluded from the shortest diagonal computation. This problem is easily fixed by performing a clean-up operation on each region before it is fed to Zhu's algorithm. Consider, for instance, a region $R(u)$, where u is horizontal and $R(u)$ lies above u. We sequentially scan the list of vertices of $R(u)$ in clockwise order, starting from the first polygon vertex following the left endpoint of $b(u)$. We remove each artificial vertex from the list, thus making its predecessor and successor into neighbors. It is easily seen that this operation preserves the monotone property of the region, and removes all artificial vertices of $R(u)$, leaving precisely the vertices of P. The end result is a monotone chain, and we complete it into a (unbounded) polygon by adding a vertex at negative infinity; the last vertex effectively replaces the edge $b(u)$. We invoke the algorithm of Zhu on this modified polygon.

The total size of all regions in the partition is $O(n)$, since each vertex appears in a constant number of regions, and each base edge without a polygon vertex at either end can be charged to its conformer vertex. It follows that it takes $O(n)$ time to compute shortest diagonals in all the regions. The main contribution of this section is an algorithm for finding a shortest diagonal whose endpoints lie in different regions of the partition. Our algorithm breaks up the problem into several instances of computing a shortest diagonal between two disjoint chains. The decomposition makes critical use of Lemma 3.3. For each node $u \in T$, we solve an instance of the two-chain shortest diagonal problem. The vertices of the two chains belong to $Z_2(u)$. In the following, we describe how these subproblems are created.

Consider an arbitrary node $u \in T$, and suppose that $v_1, v_2, \ldots, v_k$ are the children of u in T. Corresponding to each window $b(v_i)$, we create two chains of vertices: $U(v_i)$ and $V(v_i)$; the two chains are disjoint and are separated by the window $b(v_i)$. The chain $U(v_i)$ consists of just the vertices of $Z_1(v_i)$, and the chain $V(v_i)$ forms a subset of $Z_2(u) - Z_1(v_i)$.

The process of defining the chains U and V depends on the geometry of the polygon $R(u)$. In order to fix the concepts, let us assume that the generator $b(u)$ is horizontal and the region $R(u)$ lies above u. All windows $b(v_i)$ are vertical, for $i = 1, 2, \ldots, k$.

We consider a horizontal trapezoidation of $R(u)$, which is obtained by drawing maximal horizontal chords from each vertex in both directions. The top and bottom edges of each trapezoid are horizontal chords, and the left and right edges are portions of edges or windows of $R(u)$. (In a degenerate case, a trapezoid might become a triangle, with only a bottom horizontal edge.)

The family of trapezoids in this trapezoidation forms a partial order under the relation "below." In particular, a trapezoid τ is *below* another trapezoid τ' if there is a vertical line intersecting both τ and τ', and the intersection with τ lies below the intersection with τ'. If τ is below τ', then we also say that τ' is *above* τ. We now extend this partial order to windows of $R(u)$. We say that a window b *adjoins* a trapezoid τ if the left or the right boundary of τ is defined by (a portion of) b. We say that a window b is below another window b' if the *lowest* trapezoid adjoining b is below the *lowest* trapezoid adjoining b'. Finally, we say that a vertex p is above a trapezoid τ if the vertical line through p intersects τ below p. By extension, a vertex p is above a window b if p is above the lowest trapezoid adjoining b.

Lastly, we need to distinguish between windows that form the left boundary of $R(u)$ and those that form the right boundary. We call a window b a *left* (resp. *right*) window of $R(u)$ if the domain $P(b)$ lies locally to the left (resp. right) of b. With these definitions in place, let us now discuss how the set $V(v_i)$ is defined for a child node v_i. The vertices of $V(v_i)$ come either from $R(u)$ or from $Z_1(v_j)$, where $j \neq i$. The membership rules of $V(v_i)$ are as follows.

Membership Rules for $V(v_i)$:

1. A vertex $p \in R(u)$ belongs to $V(v_i)$ if $b(v_i)$ is the highest left window or the highest right window below p.
2. All vertices of $Z_1(v_j)$, for $j \neq i$, belong to $V(v_i)$ if $b(v_i)$ is below $b(v_j)$ and there is a trapezoid adjoining both $b(v_i)$ and $b(v_j)$.
3. If $b(v_i)$ is the leftmost window or the rightmost window, then $V(v_i) = Z_2(u) - Z_1(v_i)$.

The following pseudocode gives the main steps of our shortest diagonal algorithm.

DIAGONAL (P)

(* *Compute a shortest diagonal of P.* *)

1. Compute a shortest diagonal defined by a 2-neighbor pair.
2. Compute a conforming partition $\Pi(P)$, and its dual tree T.
3. **for** each node $u \in T$ **do**
4.   Compute a shortest diagonal in $R(u)$.
5.   **for** each child v_i of u **do**
6.   Compute the sets $U(v_i)$ and $V(v_i)$, and compute the shortest diagonal between disjoint chains defined by $U(v_i)$ and $V(v_i)$.

Lemma 3.3 ensures that if (p, q) is a shortest diagonal, then $p, q \in Z_2(u)$ for some node $u \in T$. If p and q both lie in $R(u)$, then the diagonal (p, q) is computed in Step 4. Otherwise, Step 6 computes the diagonal; the following lemma proves the correctness of this step.

Lemma 4.2 *Let (p, q) be a shortest diagonal of P, and let $u \in T$ be the lowest node such that $p, q \in Z_2(u)$. If at least one of p and q (say p) lies outside $R(u)$, then there is a child v_i of u such that $p \in U(v_i)$ and $q \in V(v_i)$.*

Proof: Since u is the lowest node with $p, q \in Z_2(u)$, the diagonal (p, q) intersects $R(u)$. Without loss of generality, assume that p lies outside $R(u)$, and that it lies in the domain of a left window, say $b(v_i)$. Since $p \in Z_2(u)$, it follows that $p \in Z_1(v_i)$. If v_i is the leftmost window, then the lemma clearly holds: p lies in $U(v_i)$, and $V(v_i)$ includes all the vertices of $Z_2(u) - Z_1(v_i)$ (cf. rule (3)). So, in the remainder of this proof, assume that v_i is not the leftmost window.

First, suppose that $q \in R(u)$. In this case, we claim that $q \in V(v_i)$. The claim depends on the membership rule (1) and Lemma 3.2: if $q \notin V(v_j)$, then there is window between $b(v_i)$ and q having the same orientation as $b(v_i)$. The lower endpoint of this window is necessarily a (reflex) vertex and an orthogonal coordinate system at that vertex satisfies the condition of Lemma 3.2, contradicting the assumption that (p, q) is a shortest diagonal.

If $q \notin R(u)$, then it lies in the domain of a right window, say v_j, and we must have $q \in Z_1(v_j)$. If $q \notin V(v_i)$, then $Z_1(v_j)$ must have failed the membership tests of rule (2). Since p and q are mutually visible, windows $b(v_i)$ and $b(v_j)$ cannot be incomparable; one must be above the other in the partial order. If $b(v_i)$ and $b(v_j)$ both adjoin a common trapezoid, then $q \notin V(v_i)$ implies that $b(v_i)$ is *above* $b(v_j)$. But then the symmetry of rule (2) ensures that $q \in U(v_j)$ and $p \in V(v_j)$, which proves the lemma. If $b(v_i)$ and $b(v_j)$ do not adjoin a common trapezoid, then an orthogonal coordinate system placed at the lower endpoint of the *higher* window satisfies the conditions of Lemma 3.2, contradicting the assumption that (p, q) is a shortest diagonal. This completes the proof of the lemma. $\square$

The problem of computing a shortest diagonal (p, q) with $p \in U(v_i)$ and $q \in V(v_i)$ is an instance of the problem of finding a closest visible vertex pair between two disjoint chains. Each of these problems can be solved in linear time using the following result from our paper on matrix searching with the shortest path metric.

Lemma 4.3 ([4]) *Let U and V be two vertex-disjoint chains that partition the vertices of a simple polygon P. Then a closest visible vertex pair of U and V can be found in linear time.*

Remark: In order to apply Lemma 4.3, we also need to connect the vertices of $U(v_i)$ and $V(v_i)$ into polygonal chains. The vertices of $U(v_i)$ are already joined in a polygonal chain: the boundary of the regions associated with v_i and its children. The chain for $V(v_i)$ involves concatenating some parts of $R(u)$ and $Z_1(v_j)$ for some children nodes v_j. This step also can be done easily in time proportional to $|V(v_i)|$, by first computing the horizontal trapezoidation of $R(u)$.

The following lemma bounds the total size of all the subproblems associated with a node $u \in T$.

Lemma 4.4 *Let $u \in T$ be a node, and let $v_1, v_2, \ldots, v_k$ be the children of u. The sum of the sizes of all shortest diagonal subproblems at node u is*

$$\sum_{i=1}^{k} (|U(v_i)| + |V(v_i)|) \; = \; O(|Z_2(u)|).$$

Proof: The lemma depends on the observation that all vertices in the subproblems created at u belong to $Z_2(u)$, and each vertex appears in $O(1)$ subproblems. A vertex of $R(u)$ appears in at most four subproblems: one for the leftmost window, one for the rightmost window, and at most two subproblems by rule (1). The remaining vertices of $Z_2(u)$ appear in at most four subproblems: once in a U set, once in a V set, and twice more in the sets from rule (3). This completes the proof. $\Box$

By Lemma 4.4, the total size of all the subproblems created during the algorithm *DIAGONAL* is $O(n)$. Each subproblem is solved in time proportional to its size; Step 4 uses the linear-time algorithm of Zhu [10] to compute a shortest diagonal in a monotone region; Step 6 uses the linear-time algorithm of Lemma 4.3 to compute a shortest diagonal between two disjoint chains. The only ingredient missing from the description of our algorithm is the construction of a conforming partition, which we now address.

5 Computing a conforming partition

Our partition is closely related to *window partition*, a concept useful in link distance problems [8, 9], except that we restrict the directions of visibility to horizontal and vertical. We use the common term *orthogonal visibility* to denote horizontal or vertical visibility, as the case may be. More precisely, an orthogonal visibility polygon denotes a vertical visibility polygon (the region vertically visible from the base) if the base is horizontal, and a horizontal visibility polygon if the base is vertical. If we forego the conformity condition, then a recursive partition based on orthogonal visibility meets the two most important properties demanded by our partition: each region is monotone and the windows are orthogonal to the generator. However, the regions of this partition may be non-conforming. Unlike the conventional visibility polygon, an orthogonal visibility polygon may not contain a vertex in it, which can lead to a partition of unbounded depth: consider the example of a thin parallelogram oriented at 45^o; its orthogonal partition can go on for an arbitrary number of steps.

A slight modification of the simple-minded orthogonal visibility partition delivers a conforming partition. The modification affects only those generators that do not have a vertex endpoint. Without loss of generality, assume that we start our recursive partition from a horizontal edge. Consider a generator b at some stage of the partitioning process, and let $p(b)$ denote the parent of b. If the endpoint of b that is closer to $p(b)$ is a vertex (necessarily a reflex vertex), then we set $R(b)$ to be the orthogonal visibility polygon of b, and continue the process. But if the endpoint of b is not a vertex, we perform a *conforming* *operation*. The conforming operation determines a conformed base b' corresponding to b, as follows. We sweep a segment starting from b in the domain $P(b)$, keeping the segment parallel to b and with its endpoints on the same edges as b, until the segment hits some vertex $v(b)$. Define b' to be the position of the sweeping segment at $v(b)$. Then the following two conditions hold:

1. both b and b' have their endpoints on the same edges of P, and

2. the trapezoid formed between b and b' does not contain any vertex of P in its interior.

We call the vertex $v(b)$ the *conformer vertex* of b. We set the region $R(b)$ to be the union of the trapezoid formed between b and b', and the orthogonal visibility polygon of the conformed base b'. It is easy to see that the region $R(b)$ is conforming: the conformed base b' separates $p(b)$ from all the windows of $R(b)$. Each region is also a connected, monotone polygon: a line orthogonal to base b intersects $R(b)$ in a connected set. See Figure 3 for an example of a conforming operation and a conforming partition. We have established the following lemma.

Lemma 5.1 *The recursive orthogonal visibility partition with the conforming operation gives a conforming partition of P. The total size of the partition is linear in the size of the polygon, and the depth of the partition tree is at worst linear in the number of vertices in the polygon.*

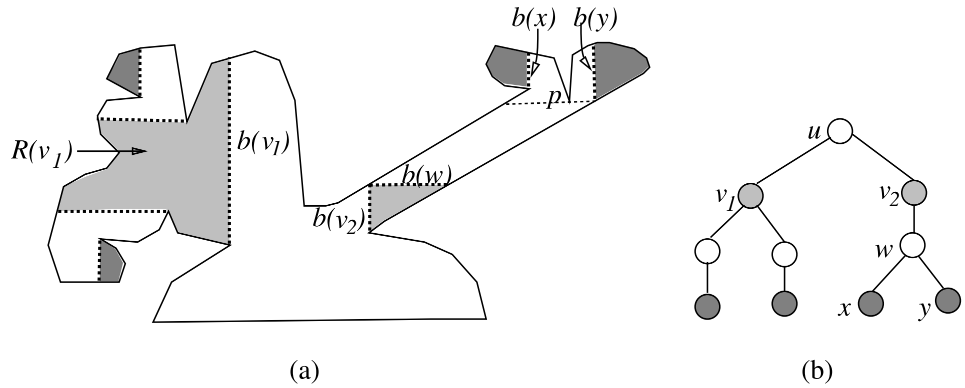

Figure 3: A conforming partition of P, and its dual tree T. Regions, and their corresponding nodes in T, are shown shaded at even levels of recursion. The conforming operation is invoked at generators $b(w)$ and $b(y)$. The conforming vertex of $b(w)$ is p, and its conforming base $b'(w)$ is the dotted chord passing through p. The remaining generators have vertex endpoints and their orthogonal visibility polygons themselves are conforming.

We can compute the partition described above in linear time, by first computing a horizontal and a vertical trapezoidation of the polygon. Given these trapezoidations, we can build our partition recursively, spending time proportional to $|R(b)|$ at each base b.

Lemma 5.2 *A conforming visibility partition of P can be computed in $O(n)$ time.*

The preceding discussion has established the main result of this section, which is summarized in the following theorem.

Theorem 5.3 *Given a simple polygon on n vertices, we can compute a shortest diagonal of the polygon in $O(n)$ time.*

Theorem 5.3 also implies a linear-time algorithm for computing a closest visible vertex pair of the polygon: a closest visible pair either forms a shortest diagonal or it is an edge of the polygon. Since the number of edges is only n, we can find a closest visible vertex pair in $O(n)$ time.

References

[1] A. Aggarwal, H. Edelsbrunner, P. Raghavan, and P. Tiwari. Optimal time bounds for some proximity problems in the plane. *Information Processing Letters*, 42:55–60, 1992.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.

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

[4] J. Hershberger and S. Suri. Matrix searching with the shortest path metric. In *Proceedings of the Twenty Fifth Annual ACM Symposium on Theory of Computing*, pages 485–494, 1993.

[5] A. Lingas. Voronoi diagrams with barriers and the shortest diagonal problem. *Information Processing Letters*, 32(4):191–198, 1989.

[6] J. O'Rourke. Computational geometry column 8. *SIGACT News*, 20(4):30, 1989.

[7] F. P. Preparata and M. I. Shamos. *Computational Geometry.* Springer-Verlag, New York, 1985.

[8] S. Suri. *Minimum Link Paths in Polygons and Related Problems.* PhD thesis, The Johns Hopkins University, 1987.

[9] S. Suri. On some link distance problems in a simple polygon. *IEEE Transactions on Robotics and Automation*, 6:108–113, 1990.

[10] B. Zhu. Computing the shortest diagonal of a monotone polygon in linear time. *Information Processing Letters*, 42:303–307, 1992.