Discrete Comput Geom 8: 131–152 (1992)

Discrete & Computational
Geometry

An $O(n \log n)$ Algorithm for Computing the Link Center of a Simple Polygon*

Hristo N. Djidjev,[1] Andrzej Lingas,[2] and Jörg-Rüdiger Sack[3]

[1] Department of Computer Science, Rice University,
P.O. Box 1892, Houston, TX 77251, USA
hristo@cs.rice.edu

[2] Department of Computer Science, Lund University,
Box 118, S-22100 Lund, Sweden
andrzej@dna.lth.se

[3] School of Computer Science, Carleton University,
Ottawa, Ontario K1S 5B6, Canada
sack@scs.carleton.ca

Abstract. We present an algorithm that determines the link center of a simple n-vertex polygon P in $O(n \log n)$ time. The link center of a simple polygon is the set of points x inside P at which the maximal link-distance from x to any other point in P is minimized. The link distance between two points x and y inside P is defined to be the smallest number of straight edges in a polygonal path inside P connecting x and y. Using our algorithm we also obtain an $O(n \log n)$-time solution to the problem of determining the link radius of P. The link radius of P is the maximum link distance from a point in the link center to any vertex of P. Both results are improvements over the $O(n^2)$ bounds previously established for these problems.

1. Introduction

This paper describes an efficient algorithm for determining the *link center* of a simple polygon. The link center was first introduced by Lenhart *et al.* [10]; we recall the definition. A *path* between a pair of points v and w in a simple polygon P is a polygonal line inside P connecting v and w. The *link distance* $d_1(v, w)$ between v and w is the minimum number of segments (straight edges) required to connect

* The research of J.-R. Sack was supported by the Natural Sciences and Engineering Research Council of Canada.

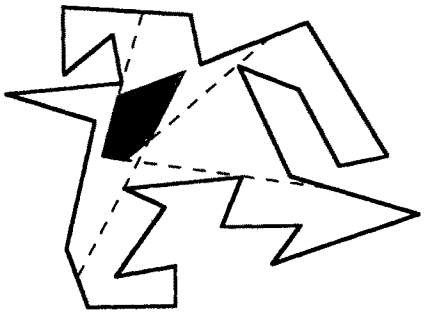

Fig. 1. A polygon P and its link center.

v to w. The *link center* is now defined as the set of all points c in P for which the maximum link distance between c and any point in P is minimized; it is denoted by LC. See Fig. 1 for an illustration of the link center.

The link-center problem has several potential applications. It could be applied to locating a transmitter so that the maximum number of retransmissions needed to reach any point in a polygonal region is minimized, or to choosing the best location for a mobile unit minimizing the minimum number of turns needed to reach any point in a polygonal region.

Suri [15] first introduced the notion of link distance. As a result of intensive research on link-distance problems and on the related visibility problems (e.g., [8], [14], and [16]), several new upper bounds have been established. In particular, a linear-time algorithm for computing the link distance between any pair of points in a triangulated polygon [15] has been developed (such a triangulation can now be determined in linear time by the recent result of Chazelle [5]). Suri [16] also gave an $O(n \log n)$-time algorithm for calculating the link diameter of a simple n-vertex polygon P. Here the *link diameter* of P is the maximum link distance between any two points of P; the link diameter is realized between two vertices of P. It is denoted by D.

As yet, no efficient algorithm for the problem of finding the link center of a simple polygon has been designed. The best algorithm currently known for finding the link center has a worst-case time complexity of $O(n^2)$ [10]. In their paper, Lenhart *et al.* [10] pose the problem of finding the link center (or at least one point inside) in subquadratic time. They also discuss the related problem of determining the link radius of a simple polygon. The *link radius* of P is the maximum link distance from a point in the link center to any vertex of P; it is denoted by R. The approach discussed in [10] for determining the link radius has a worst-case run-time of $\Theta(n^2)$.

In contrast, recall that the *geodesic distance* is the minimum (Euclidean) length of a path between two points and a *minimum geodesic path* is the shortest path connecting the points. A minimum link path and the minimum geodesic path connecting the same pair of points may be quite different as illustrated in Fig. 2. The notion of *geodesic center* is defined analogously to that of link center. The problem of finding the geodesic center has been solved in $O(n \log n)$ time in [12].

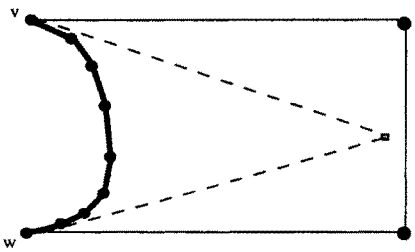

Fig. 2. A minimum link path (shaded) and the distinct minimum geodesic path (bold) connecting the vertices v and w.

Solving link-distance problems seems to be more difficult than solving the corresponding problems using the geodesic distance measure. The difficulties stem from the fact that several minimum link paths may exist connecting a given pair of vertices, while the minimum geodesic path is always unique.

In this paper we present an $O(n \log n)$ algorithm that finds the link center of any n-vertex simple polygon P. By applying our algorithm, we also obtain an $O(n \log n)$-time solution to the problem of determining the link radius of P. A preliminary version of this paper has been presented in [7]. An alternate method has been developed by Ke [9].

This paper is organized as follows. In Section 2 we give some relevant background. In Section 3 we present an $O(n \log n)$-time algorithm which determines a diagonal, called central diagonal, of P. (A *diagonal* of P is a segment inside P connecting two vertices of P.) It is shown that the link center is located in the vicinity of a central diagonal. In Section 4 we characterize regions whose intersection produces the link center and we describe the global structure of the algorithm. In Section 5 we show how to produce these regions efficiently. The knowledge of a central diagonal will aid in this. In Section 6 we describe the efficient computation of the link center as an intersection of these regions. Section 7 gives some extensions and further applications of our techniques.

2. Relevant Background and Notation

2.1. *Link Radius*

We first discuss the relationship between the problems of determining the link center and the link radius of a simple polygon. Let P be an n-vertex polygon. (Unless otherwise stated all polygons are simple and specified as a list of their vertices in counterclockwise order with no three consecutive collinear vertices.) The relation between the link radius and the link diameter of P is based on the following theorem.

Theorem 2.1 [10]. *For any polygon P, the link radius R of P is either $\lceil D/2 \rceil$ or $\lceil D/2 \rceil + 1$, where D is the link diameter of P.*

The link diameter can be found in $O(n \log n)$ time by a result in [16]. Theorem 2.1 implies that any $O(n \log n)$ algorithm for constructing the link center which is based on knowledge of the *exact* value of R, i.e., the algorithm returns the empty set, can be used to find the link center in $O(n \log n)$ time (this has also been observed in [10]). In order to do this, first apply such an algorithm on P assuming that $R = \lceil D/2 \rceil$. If R is equal to $\lceil D/2 \rceil$ the algorithm produces the (nonempty) link center. Otherwise, apply the algorithm again, this time with $R = \lceil D/2 \rceil + 1$.

Corollary 2.2. *The $O(n \log n)$ algorithm for finding the link center presented here implies an $O(n \log n)$-time algorithm for determining the link radius of an n-vertex polygon.*

2.2. Notation

In this paper the following notation is used. The segment inside P joining two points a and b is denoted by segment(a, b) (or (a, b) for short). The *visibility region* of a point q in P is the set of all points z in P which are *visible from* q, i.e., for which segment(q, z) is contained in P. The boundary of the visibility region of q defines a polygon within P called the *visibility polygon* $Vis(q, P)$ of q. For a given segment s and a point q in P, q is said to be *visible* from s if there exists at least one point z of s such that q is visible from z. The *visibility region* of a segment s in P consists of all points q in P visible from s; the boundary of the visibility region forms a polygon called *visibility polygon* $Vis(s, P)$. As a generalization we define $VisPol(d, i, X)$ to be the polygon containing all points of a polygon X at link distance at most i from d. Unless otherwise stated, $VisPol(d, i)$ stands for $VisPol(d, i, P)$.

2.3. Techniques Used

We use the following techniques developed for solving visibility problems inside any triangulated simple n-vertex polygon P:

1. For any segment s in P, $Vis(s, P)$ can be computed in $O(n)$ time [8], [17].
2. Given a fixed edge e of P, P can be preprocessed in $O(n)$ time so that, given any point x inside P, the subsegment of e of the points visible from x can be constructed in $O(\log n)$ time [8], [13].
3. A triangulated polygon P can be preprocessed in $O(n)$ time so that, given any point x inside P and any direction u, the first point on the boundary of P hit by the ray in direction u from x can be computed in $O(\log n)$ time [8]. Such an operation is called a *shooting query* and the first segment of the ray inside P is called the *shot*.

3. Locating a Region Containing the Link Center

In this section we characterize a "small" region inside a given n-vertex polygon P which contains the link center of P. We can determine this region in $O(n \log n)$

time. The region serves as an approximate location of the link center and guides the subsequent efficient search for the precise location of the link center.

3.1. *Notation*

We introduce some important notation. Let s be any segment in P. Then the *covering radius* $CovRad(s, P)$ of s is defined by

$$CovRad(s, P) = \max_{p \in P} \min_{q \in s} d_L(q, p).$$

Let d be any edge of a given triangulation of P. For the remainder of this section we denote the polygons defined by the partitioning of P induced by d as $P_1(d)$ and $P_2(d)$. The *covering difference* of d in P is the value $|CovRad(d, P_1(d)) - CovRad(d, P_2(d))|$.

A diagonal with minimum covering difference among all diagonals induced by the given triangulation is called a *central diagonal.*

The *covering radius pair* of d, $CR(d)$, is the pair of covering radii of d in $P_1(d)$ and $P_2(d)$. (When no ambiguity arises we use P_1, P_2, and CR instead of $P_1(d)$, $P_2(d)$, and $CR(d)$, respectively.)

3.2. *Determining a Central Diagonal*

We first describe the location of the link center with respect to an arbitrary diagonal of P. Then we give an algorithm to find a central diagonal d of P and show that the link distance from d to any point of the link center is no more than two. Thus the region $VisPol(d, 2, P)$ contains the link center and provides an approximate location of the link center.

Lemma 3.1. *Let d' be a diagonal of P for which $CR(d') = (c_1, c_2)$, $c_1 \geq c_2$. Then $VisPol(d', R - c_2 + 1, P)$ contains the link center of P.*

Proof. Let d' be the diagonal with $CR(d') = (c_1, c_2)$, $c_1 \geq c_2$. Let P_1, P_2 be the subpolygons induced by d' such that $CovRad(d', P_i) = c_i$, $i = 1, 2$. Assume that q is a point of the link center of P that does not belong to $VisPol(d', R - c_2 + 1, P)$. We first assume that P_1 is that polygon among P_1 and P_2 which contains q. Since $CovRad(d', P_2) = c_2$, there exists a point v of P_2 such that $d_L(z, v) \geq c_2$ for each z on d'. Now construct a minimum link path $p = (p_1, \ldots, p_k)$ from $q = p_1$ to $v = p_k$. Since q is in P_1 the path p intersects d', say at point z'. Furthermore, q is not in $VisPol(d', R - c_2 + 1, P)$ and thus z' is on segment (p_i, p_{i+1}), for $i > R - c_2 + 1$. From $d_L(z', v) \geq c_2$ it follows that $k \geq i + c_2 - 1 > (R - c_2 + 1) + c_2 - 1 \geq R + 1$. Hence q does not belong to the link center of P.

Analogously, if we assume that P_2 contains q, there exists a point v of P_1 for which $d_L(q, v) > (R - c_2 + 1) + c_1 - 1 \geq R + 1$ (The latter inequality holds since by assumption $c_1 \geq c_2$.) □

Lemma 3.2. *The covering difference of a central diagonal is at most one.*

Proof. Assume by contradiction that the covering difference for all edges of a given triangulation of the simple polygon P is at least two. Let Δ be a triangle of the triangulation containing at least one point, p, of the link center, denoted by LC. Denote the covering radii of the edges of Δ as per Fig. 3. Since p is in LC it follows that a_1, b_1, and c_1 are no larger than R. Since R is the link radius at least one of these covering radii is at least $R-1$. Assume that $c_1 = \max\{a_1, b_1, c_1\}$. Two cases arise: (1) $c_1 = R-1$ and (2) $c_1 = R$.

(1) Assume that $c_1 = R-1$. By the initial assumption, $c_2 \geq R+1$ or $c_2 \leq R-3$. Consider the case $c_2 \geq R+1$. The point p is in LC and thus $c_2 = R+1$. From $c_1 = \max\{a_1, b_1, c_1\}$ it follows that both a_1 and b_1 are at most $R-1$. Now, side c sees each of other sides which contradicts that $c_2 = R+1$. (The case where $c_2 \leq R-3$ is handled analogously to case (2)'below.)

(2) Assume now that $c_1 = R$. In the given triangulation of P the edge c is adjacent to another triangle Δ' whose edges are called c, f, and g. Denote by P_c, P_f, and P_g the subpolygons induced by c, f, and g, respectively, which are not containing Δ. (See Fig. 3 for an illustration.) Assume first that $CR(f)$ or $CR(g)$ equals $(R, R-2)$, without loss of generality say that $CR(f) = (R, R-2)$. Subpolygon P_c (equal to P_f, and P_g and Δ'), contains at least one more triangle than P_f, and thus to complete the proof in this case, we can apply induction on the number of triangles of P_c.

Assume now that neither $CR(f)$ nor $CR(g)$ equals $(R, R-2)$. Without loss of generality assume further that $CovRad(f, P_f) \geq CovRad(g, P_g)$. Let $CR(f) = (r_1, r_2)$, where $r_1 = CovRad(f, P_f)$ (as depicted in Fig. 3); then $r_1 \geq R-1$. It is now straightforward to check that no pair (r_1, r_2) satisfying all of the conditions $|r_1 - r_2| \geq 2$, $(r_1, r_2) \neq (R, R-2)$, $R-1 \leq r_1 \leq R$, and $R-2 \leq r_2$ exists. □

Lemma 3.3. *In any simple polygon with covering radius R there exists a central diagonal with a covering radius to either side being at least $R-1$.*

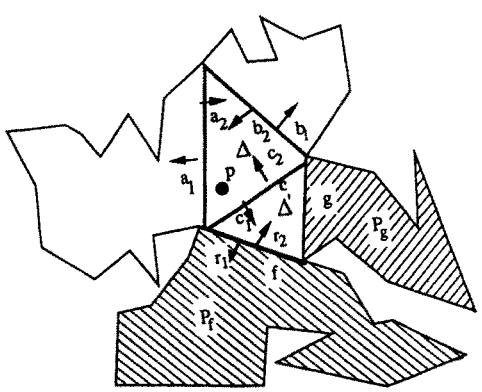

Fig. 3. Notation of Lemma 3.2.

Proof. Let d be a central diagonal of P with the covering radius pair (c_1, c_2); assume that $c_1 \geq c_2$. Then from Lemma 3.2 it follows that $c_1 - c_2 \leq 1$. Since R is the link radius neither $c_1 > R$ and $c_2 > R$ nor $c_1 \leq R - 2$ and $c_2 \leq R - 2$ are possible. Therefore it will be enough to prove that $(c_1, c_2) \neq (R - 1, R - 2)$. Although a triangulation edge d may exist with the covering radius pair $(R - 1, R - 2)$, we will show that d could not minimize the value $c_1 - c_2$. Assume the contrary, i.e., let the covering difference of any edge of the triangulation be at least one and let d be a central diagonal with the covering radius pair $(R - 1, R - 2)$. For any triangulation edge e, let $P(e)$ be the subpolygon of P determined by e with the greater covering radius from e. Let d' be a triangulation edge with the covering radius pair $(R - 1, R - 2)$ such that no other edge d'' with the covering radius pair equal to $(R - 1, R - 2)$ and $P(d'') \subset P(d')$ exists.

Assume for ease of notation that d' is vertical. Without loss of generality let the covering radii to the left be $R - 2$ and to the right be $R - 1$. Let the triangle adjacent to d' and to the right of d' have edges d', d_r, and d_b. For one of d_r or d_b, say d_r, the covering radius pair is (R_1, R_2) with $R_i \in \{R - 1, R - 2\}$, $i = 1, 2$, and, since d' minimizes the value $|c_1 - c_2|$, $R_1 \neq R_2$. Let R_1 be the covering radius of d_r with respect to the subpolygon induced by d_r that contains d' and let R_2 be the other covering radius. In case R_1 equals $R - 2$ the argument is repeated by replacing d' by d_r. Consider now the second case, $R_1 = R - 1$ and $R_2 = R - 2$. Then d_b cannot be an edge of the polygon. Otherwise, any point in the triangle would reach all other points in at most $R - 1$ links. This contradicts the definition of R. Thus d_b is an edge of the triangulation and has the covering radius pair $(R^*, R - 1)$, where R^* is either $R - 1$ or $R - 2$. In the former case we obtain a contradiction to the fact that the covering difference of any edge is at least one; the latter case contradicts the choice of d' (we should have chosen d_b instead of d'). □

Lemma 3.4. *The polygon VisPol(d, 2, P) of a central diagonal d contains the link center of the simple polygon P.*

Proof. Let d be a central diagonal of P. By Lemma 3.3, the covering radius of d to either side is at least $R - 1$. By Lemma 3.1, $VisPol(d, R - (R - 1) + 1, P) = VisPol(d, 2, P)$ contains the link center. (In case the minimum covering radius is R, $VisPol(d, 1, P)$ contains the link center.) □

The proof of Lemma 3.3 provides a method for finding a central diagonal of a simple polygon. Algorithm 3.1 given below determines a central diagonal d of P more efficiently, in $O(n \log n)$ time. The algorithm uses the dual graph of the triangulation graph of P. The *dual graph* T of a triangulation of P is constructed by associating a vertex of T for each triangle of the triangulation of P and joining a pair of vertices if their corresponding triangles share an edge of the triangulation of P. T is a binary tree and a one-to-one correspondence exists between the edges of T and the internal edges of the triangulation of P. (See [4], [8], and [10] for other algorithms based on this idea.) We use the following well-known fact stated, e.g., in [11] and [6].

Lemma 3.5. *For any binary tree T with n vertices there exists an edge of T whose removal divides T into two subtrees of no more than $2n/3 + 1$ vertices each. Such an edge can be found in $O(n)$ time.*

We now state the algorithm for finding a central diagonal of a simple polygon.

Algorithm 3.1. Finding a central diagonal
Input: A simple polygon and its triangulation T'
Output: A central diagonal d
If the triangulation T' consists of a single edge
then output this edge and **stop**
else find an edge d of the triangulation of P whose corresponding edge in T' divides T' into two subtrees T_1 and T_2 each containing no more than $2|V(T')|/3 + 1$ vertices, where $V(T')$ denotes the vertex set of T' and $|V(T')|$ denotes its cardinality.
Compute the covering radii c_1, c_2 of d and assume that $c_1 \geq c_2$.
Let T_1 be the subtree corresponding to that subpolygon induced by d whose covering radius is c_1 and let T_2 be the other subtree of T'.
If $c_2 \geq R - 1$ **then stop**
else apply the algorithm recursively for $T' = T_1$.

The correctness and the time complexity of Algorithm 3.1 are established in the following theorem.

Theorem 3.6. *In $O(n \log n)$ time Algorithm* 3.1 *finds a central diagonal d for any n-vertex simple polygon P.*

Proof. First we establish the time complexity of Algorithm 3.1. Let T be the tree corresponding to the triangulation of P. We perform Algorithm 3.1 initially letting T' equal T. Since the covering radii of an edge of a triangulation of P can be determined in linear time [16], each of the $O(\log n)$ iterations of the algorithm takes $O(n)$ time. (Note that in general P is different from the polygon associated with T'.) This yields an $O(n \log n)$-time bound on the running time of Algorithm 3.1. It remains to establish the correctness of the algorithm. Every time Algorithm 3.1 is recursively called it holds that $c_2 = \min\{c_1, c_2\} \leq R - 2$. Each diagonal of that subpolygon of P determined by c_2 has covering radius pair (c_1', c_2') where $\min\{c_1', c_2'\} \leq R - 2$. By Lemma 3.3 such a diagonal cannot be a central diagonal. Thus the portion of P corresponding to T_2 does not contain a central diagonal. Consequently, Algorithm 3.1 correctly examines only T_1 for any subsequent iteration. □

Since the visibility polygon from a segment can be computed in linear time, Corollary 3.7 follows by Lemma 3.4 and Theorem 3.6.

Corollary 3.7. *In $O(n \log n)$ time a diagonal d can be identified such that the link center is contained in the polygon VisPol(d, 2, P).*

4. The Algorithm

As we recall, in Section 4.1, Lenhart *et al.* show that the link center of P can be computed as the intersection of a particular set of subpolygons, called neighborhood polygons, of P. The total size of these polygons does not allow for an efficient, i.e., $O(n \log n)$, solution to compute this intersection. Thus, in Section 4.2 we define a smaller set of subpolygons whose intersection still gives the link center, but which can be identified and intersected efficiently. To accomplish this we use the central diagonal obtained in Section 3 which provides us with an approximate location of the link center. In Section 4.3 we give an outline of the entire algorithm to compute the link center. The details of the algorithm are subsequently discussed in Section 5.

4.1. *Neighborhood Polygons*

We restate some notation and two facts from [10]. The *k-neighborhood* about a point z in P is defined by

$$N_k(z) = \{z' \in P \mid d_L(z, z') \leq k\}.$$

Fact 4.1 [10]. Let $i \geq 1$ and let v be a point of P. The $N_i(v)$ region is a subpolygon of P whose vertices all lie on the boundary of P and whose edges are of the following two types:

(1) a (portion of an) edge of P;
(2) a segment (r, z) connecting a reflex vertex r of P to a point z on the boundary of P.

Fact 4.2 [10]. The link center of a polygon P is the intersection of the sets $N_R(v)$ over all convex vertices v of P, where R is the link radius of P.

From Lemma 3.4 it follows that to compute the link center it suffices to find the intersection over all convex vertices of P at distance at least $R - 3$ from a central diagonal.

4.2. *Boundary Segment Polygons*

Assume that we have executed Algorithm 3.1 and obtained a central diagonal d of P. Using a linear-time segment-visibility algorithm (e.g., [8]) we determine the polygon *VisPol*$(d, 1)$. In linear time we can construct the window tree W as introduced by Suri [16]. We first recall the constructive definition.

On the boundary of *VisPol*$(d, 1)$, several segments, called *lids* of P, that do not belong to the boundary of P may exist. Each lid divides P into two subpolygons, the one which does not contain *VisPol*$(d, 1)$ is called a *pocket* of P. In each of

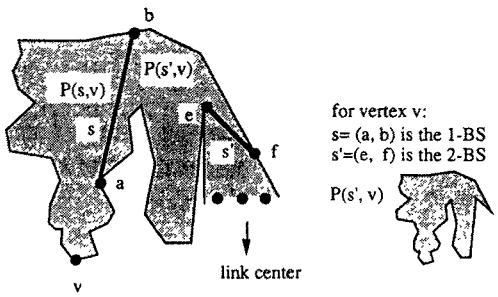

Fig. 4. Illustration of definitions of i-BS for $i = 1, 2$ and the corresponding i-neighborhood polygons.

these pockets we recursively compute the polygon visible from the respective lid. Now we associate with each of these visibility polygons a node of a tree W, where two nodes are connected by an arc if and only if their associated polygons share a lid. We define the node corresponding to $VisPol(d, 1)$ to be the root of W. This tree W is the *window tree* of d. To simplify the notation we identify the node of W representing a polygon by the polygon itself. The polygons (associated with nodes) of W are called *regions* of P.

Let r be a reflex vertex of P and let (r, z_1), (r, z_2) be two edges of some $N_i(v)$ region. We refer to (r, z_1) as the *clockwise N_i-boundary segment* of r if the vertex r appears between z_2 and z_1 on a clockwise traversal of the boundary of the corresponding N_i region starting at z_2. The edge (r, z_2) is then referred to as the *counterclockwise N_i-boundary segment* of r. The following definitions are illustrated in Fig. 4. Let s be a boundary segment from some $N_j(v)$ region, $j \geq 1$, partitioning P into two subpolygons. The subpolygon containing vertex v is referred to as $P(s, v)$, or simply $P(s)$ if no ambiguity arises. Let d be some (fixed) central diagonal of P. For any convex vertex v of P and edge s on the boundary of $N_i(v)$, s is called an *i-boundary segment*, an i-BS for short, (for v) if

(a) $i < R$ and v lies in a subpolygon of P induced by s different from $VisPol(d, 2)$, or

(b) $i \leq R$ and s belongs to $VisPol(d, 2)$.

For the computation of the link center we use some i-BSs and the polygons $P(s)$ instead of the regions $N_i(v)$. If s is an i-BS for v, then $P(s, v)$ is called the i-BS *polygon* for v induced by s. It is clear from the above definitions that $P(s, v) \supseteq N_i(v)$. We refer to the i-BSs, for the same i, as BSs of the same *generation*. If for some v and i there is a single i-BS for v, then the i-BS is denoted by $b_i(v)$ and the i-BS polygon is denoted by $P(b_i(v))$. To simplify the notations we define $b_0(v) = v$ and $P(v) = v$ for any vertex v.

From the definitions of link center and N_k-region Corollary 4.3 follows.

Corollary 4.3. *Let k be an integer in $[0, R]$. A point z belongs to the link center of a polygon P if and only if the link distance between z and all polygons $N_k(v)$ is $\leq R - k$, for all convex vertices v.*

To gain in efficiency compared with the $O(n^2)$ link-center algorithm of [10] we

use the k-BS polygons $P(b_k(v))$ instead of the neighborhood regions $N_k(v)$. Theorem 4.5 stated below justifies this approach.

First we argue that certain BSs are irrelevant for the computation of the link center. Assume that two k-BSs of the same generation are incident to the same reflex vertex r. Then one of the corresponding k-BS polygons will contain the other and the one contained can be removed from further consideration (Fact 4.2 and the definition of i-BS). More generally, for any two segments of the same generation for which one of the corresponding polygons contains the other, the same argument holds (i.e., one can be removed). Assume now that two BSs are incident to a common reflex vertex, but they are from different generations. As before, the lower generation BS can be removed since the polygon corresponding to its BS successor will always contain that polygon corresponding to the higher generation BS.

Formally, we define a partial order on the set of all BSs as

$$s_1 \geq s_2 \qquad \text{if } s_1 \text{ and } s_2 \text{ are BSs for which } \quad P(S_1) \supseteq P(S_2).$$

The minimal elements with respect to this partial order are called *relevant* BSs and all other BSs are called *irrelevant.*

Lemma 4.4. *Let k be an integer in $[0, R]$. A point z belongs to the link center of a polygon P if and only if the link distance between z and the polygons $P(s)$ is $\leq R - k$ for any relevant k-BS s.*

Proof. ($\Rightarrow$) Follows from Corollary 4.3 and the fact that $P(b_i(v)) \supseteq N_i(v)$.

($\Leftarrow$) The result is established by induction on k.

Let $k = 0$. Since the link distance between a point z in the link center and any vertex v in P is at most R the result follows in this case by having defined $P(b_0(v))$ to be v. For the induction let k be chosen so that $0 \leq k < R$. Assume that if the link distance between some point z^* and the polygon $P(s)$ is $\leq R - k$ for any relevant k-BS s, then z^* belongs to the link center of P. Assume now that the link distance between some point z and the polygon $P(s)$ is $\leq R - k - 1$ for any relevant $(k + 1)$-BS s. Consider an arbitrary relevant k-BS s'. By the definition of a BS and a relevant BS there exists a relevant $(k + 1)$-BS s'' such that

$$\max_{z'' \in P(s'')} \min_{z' \in P(s')} d_L(z', z'') \leq 1.$$

Then by the last assumption the link distance between z and the polygon $P(s')$ is $\leq R - k$. Since s' was chosen arbitrarily, the link distance between z and the polygon $P(s)$ is $\leq R - k$ for any relevant k-BS s. Then from the inductive assumption it follows that z belongs to the link center. □

It now follows:

Theorem 4.5. *The link center of a polygon P is the intersection of $VisPol(d, 2)$ with the sets $P(b_R(v))$ over relevant $b_R(v)$ for all convex vertices v of P, where R is the link radius of P.*

4.3. *Outline of the Algorithm*

In Algorithm 4.1 we outline the computation of the link center of P; the algorithm is shown to take $O(n \log n)$ time. Where indicated in the algorithm some steps are discussed in detail in Sections 5 and 6.

Algorithm 4.1. Outline of the algorithm for finding the link center
Input: A triangulated simple polygon P
Output: The link center of P

1. Find the link diameter D by the algorithm from [16]. Choose $R := \lceil D/2 \rceil$ (the smaller of the two possible values given by Theorem 2.1).
2. Find a central diagonal d by applying Algorithm 3.1.
3. Construct the window tree of P with respect to d in $O(n)$ time [16].
4. Produce a postorder numbering (see, e.g., [1]) of the vertices of the window tree.
   Let $R_1, \ldots, R_j$ be the ordering of the regions of P corresponding to the numbering of the vertices of the window tree. R_j is the visibility polygon from d and $R_k, R_{k+1}, \ldots, R_j - 1$ are the children of R_j. (Notice that according to this ordering all children of a region R_i will precede R_i in the region sequence. (See Fig. 5.))

5.1. **For** $i := 1$ **to** $k - 1$ apply Algorithm 5.1 given below to find all relevant BSs, s, incident with reflex vertices in R_i, so that $P(s)$ does not intersect $R_k \cup \cdots \cup R_j$.

5.2. **For** $i := k$ **to** j apply Algorithm 5.1 given below to find all relevant BSs, s, incident with reflex vertices in R_i.
   {Steps 5.1 and 5.2 specify the order in which to compute the BSs.}

6. Find the intersection of the boundary polygons determined by the relevant R-BSs using Algorithm 6.1.
7. **If** the intersection region computed in Step 6 is empty **then** repeat Steps 2–6 choosing $R := \lceil D/2 \rceil + 1$ (the other possible value of R).

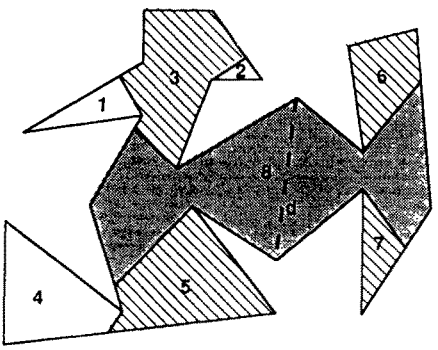

Fig. 5. A possible ordering of the regions for Step 4.

5. Determining i-Boundary Segments Efficiently

In this section we discuss the efficient computation of all i-BSs. For this consider an arbitrary, but fixed, region U of the window tree with lid l together with the children polygons of U. Call the entire polygon so formed U^+. Therefore U^+ is that portion of the $VisPol(l, 2, P)$ that lies on the same side of l as does U. Assume that all children of the node corresponding to U have been processed (inductively). Let $i-1$ be the maximum index on the BSs produced so far that intersects U^+ (see Algorithm 4.1). It is easy to notice that any BS in U^+ of generation $i-4$ or below can be omitted from consideration. We assume first that all relevant BSs in U^+ are relevant $(i-1)$-BSs leaving the generalization to $(i-2)$, $(i-3)$-BSs to the end of this section.

We describe an algorithm to compute all relevant i-BSs incident to reflex vertices of U that are of type (a) (as per the definition of i-BS given in Section 4.2). The algorithm is shown to require $O(|U^+| \log |U^+|)$ time. Each reflex vertex in P is examined $O(1)$ times. Thus the total time taken to find all i-BSs (counterclockwise and clockwise) for each reflex vertex in P is bounded by $O(n \log n)$. Without loss of generality we discuss only the determination of counterclockwise i-BSs simply referred to here as i-BSs.

For any reflex vertex r in U there might be as many as $|U^+|$ relevant $(i-1)$-BSs that are visible from r. In Lemma 5.2 we first show that at most one relevant $(i-1)$-BS needs to be considered to compute the relevant i-BS incident to r (if any). In Lemma 5.3 we show how to determine the relevant i-BS incident to r given the unique relevant $(i-1)$-BS. A given $(i-1)$-BS can produce relevant i-BSs incident to several reflex vertices in U. The efficient computation of all such i-BSs is described in Algorithms 5.1–5.3.

5.1. *Determining the Unique $(i-1)$BS that Generates a Relevant i-BS and Determining the i-BS from the $(i-1)$BS*

Let $s_l = (t_l, h_l)$ be a relevant i-BS where t_l, h_l are the endpoints of an edge of some $N_i(v)$ polygon. By definition of i-BS the interior of $N_i(v)$ is to the right of s_l and s_l is oriented from t_l to h_l. Then t_l is called the tail of s_l and h_l is called its head. Assume that the $(i-1)$-BSs of U^+ are *cyclically* sorted, i.e., for $s_l = (t_l, h_l)$ and $s_j = (t_j, h_j)$, s_l is said to be *less than* s_j if h_l occurs before h_j on a counterclockwise traversal of U^+ starting at some tail. Notice that since the BSs are relevant, it is not possible that $s_l \neq s_j$ and $h_l = h_j$. Produce a sorted cyclic list.

The first $(i-1)$-BS (in the cyclic ordering) for a given reflex vertex r (where r belongs to the clockwise polygonal chain from the tail to the head of the $(i-1)$-BS) is called the *first $(i-1)$-BS for r*. (See Fig. 6 for an illustration.) We say that a relevant $(i-1)$-BS, $b_{i-1}(v)$, *contributes* to the ith generation *at* reflex vertex r if a relevant i-BS, $b_i(v)$, is incident to r.

A segment s_k may be the first $(i-1)$-BS for many reflex vertices; we denote the set of all such reflex vertices by $\mathbb{R}_k$. It is easy to see that the following observation holds.

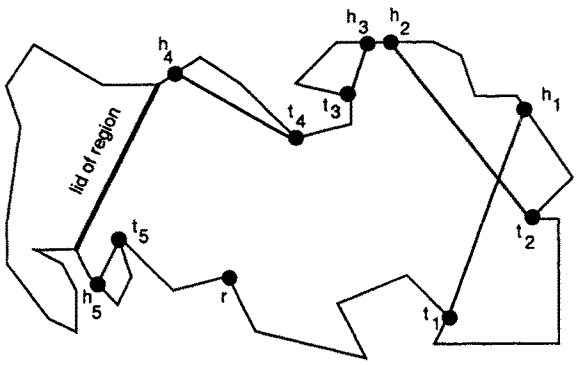

Fig. 6. A reflex vertex r and segments $s_i = (t_i, h_i)$ cyclically sorted from r. The segment $s_1 = (t_1, h_1)$ is the relevant i-BS for r.

Observation 5.1. All sets $\mathbb{R}_k$ of those reflex vertices in U for which s_k is the first $(i-1)$-BS for all relevant BSs s_k in U can be determined in $O(|U^+|)$ time. Furthermore, the sets $\mathbb{R}_k$ lie on nonoverlapping chains on the boundary of U.

As mentioned above we assume first that all BSs entering the particular region U of the window tree are $(i-1)$-BSs for some $i > 0$, i.e., they are from the same generation. For this situation, we now discuss how to compute the i-BSs. In the next two lemmas we show how the relevant i-BSs can be identified.

Note that, for $i = 1$ (*the basis of our iterative construction*), *the* 0-BSs for a vertex v equals v. We can see this as the special case of a BS which is degenerated to a single point.

Unless otherwise stated, all polygonal chains mentioned are counterclockwise. We denote the polygonal chain from a boundary point a of P to a boundary point b of P by $CH_Q(a, b)$. (The index Q is omitted when no ambiguity arises.) *Polygon*(a, b) stands for the chain $CH(a, b)$ closed by adding the segment(b, a) assuming that segment(b, a) together with $CH(a, b)$ form a simple polygon.

Lemma 5.2. *If a reflex vertex r is incident to a relevant i-BS, then r sees its first $(i-1)$-BS.*

Proof. Assume that r does not see its first $(i-1)$-BS, s, for vertex v and that the BS which contributes to the ith generation at r is $s' = b_{i-1}(w)$, $w \neq v$. Then r sees s'. Let the relevant i-BS incident to r be $b_i(w) = (r, z'')$. Let $s = (t, h)$ and $s' = (t', h')$. (See Fig. 7 for an illustration.) Now let p be a point on s' that is visible from r and has the minimum distance to h' among all such points. Assume without loss of generality that $p \neq h'$. Then there exist points z' and z on $CH(r, h')$ and $CH(h', z'')$ collinear with r, z'' and visible from r, respectively. Since s is the first $(i-1)$-BS for r, then s lies inside the counterclockwise *polygon*(r, z'). Let $b_i(v)$ be any i-BS of v. We now show that $P(b_i(w)) \supseteq P(b_i(v))$, which contradicts the assumption that $b_i(w)$ is a relevant i-BS. Assume that there exists a point $q \in P(b_i(v)) - P(b_i(w))$. Then some segment (q, f) exists interior to P such that $f \in P(b_{i-1}(v))$. The segment (q, f)

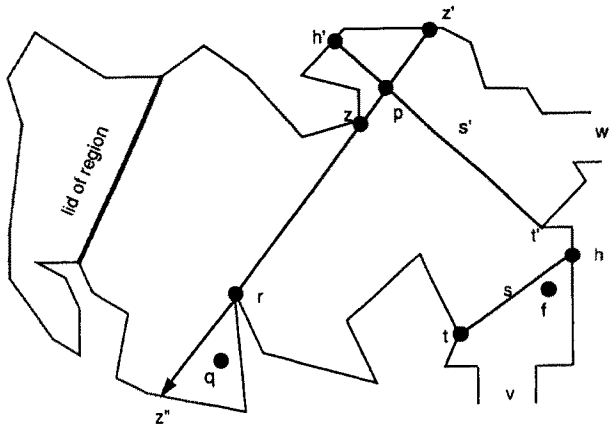

Fig. 7. Illustration of the proof of Lemma 5.2. If r is not visible from s, then no point from Polygon($z'', \ldots, r$) can be visible from s and r could not contribute to the link center.

intersects $b_i(w)$, whence q would lie in *Polygon*(z'', r). Hence, f is in *Polygon*(z, z'') and therefore f cannot be in $P(b_{i-1}(v))$ which contradicts our assumption. □

When a reflex vertex contributes to the link center its contribution can be determined using Lemma 5.3 below. To state the lemma we need some notation.

Let $s = (t, h)$ be the first $(i-1)$-BS for r. Let p be the point on s that is visible from r and has minimum Euclidean distance from h (among all such points). Then the *segment*(r, p) is called the *extremal segment* for r and s. Let (r, z) be a (directed) segment interior to a polygon connecting a reflex vertex r to a boundary point z. Then the *back-extension* for (r, z) is the longest internal segment originating at r and directed oppositely to (r, z).

Lemma 5.3. *Let $s = (t, h)$ be the first $(i-1)$-BS for r and assume that s is visible from r. Then the back-extension of the extremal segment is the i-BS incident to r (if it exists).*

Proof. Let p be the endpoint of the extremal segment for r and s (other than r). Assume that $p \neq h$. Then there exists a vertex z on (open) $CH(h, r)$ collinear with r and h. It is easy to see that the segment (r, z'') lies on the boundary of *VisPol*(s, 1). Let *segment*(r, z'') be the back-extension for segment(r, z) and let segment(p, z') be the back-extension for segment(p, z). Suppose that segment(r, z'') is not the BS for r. (See Fig. 8 for an illustration.) Then there exists another segment $s' =$ segment(t', h') producing a smaller counterclockwise angle at r. Observe first that s' does not intersect the ray originating at r toward p. Otherwise, the visibility polygon of s' would include segment(r, z'').

Thus both endpoints of s' are on one of these chains:

(a) $CH(r, z')$ which contradicts the assumption that s is the first $(i-1)$-BS for r, or

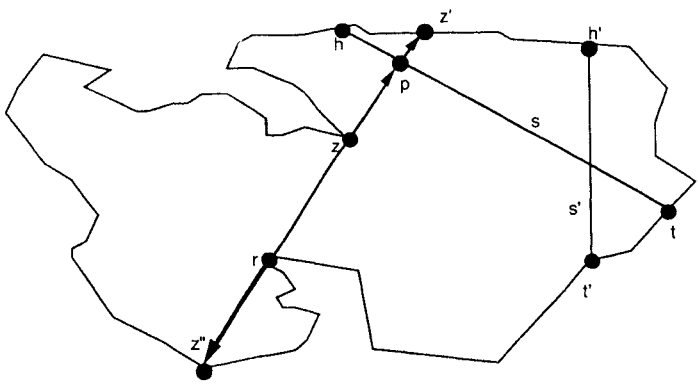

Fig. 8. Illustration of Lemma 5.3. The back-extension (r, z'') of the extremal segment (r, p).

(b) $CH(z', z)$ which means s' is not visible from r, a contradiction, or
(c) $CH(z, r)$ which contradicts the choice of angle.

The contradictions show that (r, z'') is the BS for r. □

5.2. *Computing Relevant Boundary Segments*

Let U be some arbitrary, but fixed, region of the window tree. We still assume that all BSs necessary for the computation of i-BS for reflex vertices in U are $(i-1)$-BSs of the region U^+, where U^+ is the polygon formed by U together with its children polygons in the window tree. Algorithm 5.1 below gives a sketch of the construction of all relevant i-BSs; the details are subsequently given in Algorithms 5.2 and 5.3. The output of Algorithm 5.1 is used in Algorithm 6.1, the determination of the boundary of the link center, discussed in Section 6. For ease of description, we assume each region of the window tree is retriangulated separately.

Algorithm 5.1. Construction of the relevant i-BSs
Input: A region U of the window tree; all relevant $(i-1)$-BSs of U^+ of the window tree, where U^+ is the polygon formed by U together with its children polygons
Output: All relevant i-BSs incident to reflex vertices in U generated from the $(i-1)$-BSs entering U^+
1. sort the $(i-1)$-BSs cyclically starting at some tail of a segment;
2. let $\mathbb{S} = s_1, \ldots, s_{\text{last}}$ be the sorted list so produced;
3. **for** $s_k := s_1$ **to** s_{last} **do**
   Determine the set $\mathbb{R}_k$ of all reflex vertices in U for which s_k is a relevant $(i-1)$-BS;
   for each reflex vertex r in $\mathbb{R}_k$ **do**
   compute the relevant i-BS incident to r; {details given below in Algorithms 5.2 and 5.3}

By Observation 5.1 all sets $\mathbb{R}_k$ can be determined in $O(|U^+|)$ time. In this section we address the determination of the relevant i-BSs for all reflex vertices in $\mathbb{R}_k$. By Lemma 5.3 a boundary segment for a reflex vertex r in $\mathbb{R}_k$ is the back-extension of an extremal segment for r and s_k. To compute the extremal segments for all reflex vertices r in $\mathbb{R}_k$ efficiently we construct a polygon S_k in which we compute the extremal segments. The extremal segments computed in S_k are identical to those produced in P, but are more efficiently determined. The construction is as follows.

We denote the minimum Euclidean length path inside a polygon from a point a'' to a point b'' by $path(a'', b'')$. Let (a, b) be a lid of a region U of the window tree ($(a, b) \neq d$) and let U^+ be U plus its children polygons. Let $r_f, \ldots, r_l$ be the reflex vertices in $\mathbb{R}_k$ listed in (counterclockwise) polygonal order starting at the head of the $(i-1)$-BS determining $\mathbb{R}_k$.

The polygons S_k are now constructed as hourglasses (slightly generalized) as used in [8]. More specifically, S_k is composed of the shortest paths $path(h_k, r_f)$ and $path(r_l, t_k)$, the chain $CH_{U^+}(r_f, r_l)$, and the segment s_k. The segment s_k serves as the base from which the hourglass is constructed. See Algorithm 5.2 for a description and refer to Fig. 9 for an illustration of the procedure.

In case the shortest paths $path(h_k, r_f)$ and $path(r_l, t_k)$ share one or more edges, we notice (analogously to [8]) that no reflex vertex on the chain $CH_{U^+}(r_f, r_l)$ can see its first segment s_k.

Algorithm 5.2. Construct S_k

Input: All relevant segments s_k together with their relevant reflex vertex sets $\mathbb{R}_k$

Output: All polygons S_k

for each pair $(s_k, \mathbb{R}_k)$ in polygonal order sorted by the tails of s_k **do**
  let $r_f, \ldots, r_l$ denote the chain of reflex vertices in $\mathbb{R}_k$;
  compute p_1 as $path(h_k, r_f)$;
  compute p_2 as $path(r_l, t_k)$;
  if p_1 and p_2 share one or more edges **then**
    remove S_k from consideration;
  polygon S_k is formed by connecting p_1, $CH_U(r_f, r_l)$, p_2, and the segment (t_k, h_k).

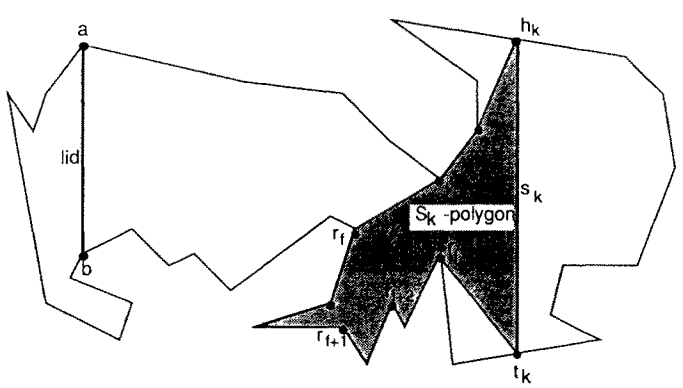

Fig. 9. Construction of S_k-polygons for efficient determination of back-extensions.

Lemma 5.4. *All polygons S_k can be constructed in total $O(n \log n)$ time.*

Proof. Let s_k be the segment corresponding to $\mathbb{R}_k$, i.e., the first segment for the reflex vertices $r_f, \ldots, r_l$. Note first that since r_f and r_l are in U and t_k and t_l are in U^+ the paths p_1 and p_2 lie in U^+. If r_1, r_2, and r_3 are reflex vertices so that r_2 is located between r_1 and r_3 counterclockwise and s_k is the first segment for r_1 and r_3, then clearly s_k is the first segment also for s_2. Thus for different $\mathbb{R}_k$ the chains $r_f, \ldots, r_l$ are disjoint; likewise are the paths $path(r_l, t_k)$. Now, any reflex vertex on p_2 may belong to at most one other polygon $S_{k'}$. For this, consider a reflex vertex r that lies on the two shortest paths, $path(h_k, r_f)$ and $path(h_{k'}, r_{f'})$. If r lies on the counterclockwise chain from h_k to r_f, then r cannot lie on another counterclockwise chain for some $h_{k'}$, $r_{f'}$. The vertex r may, however, lie on some path $path(h_{k'}, r_{f'})$ if the path lies on the clockwise chain from $r_{f'}$ to $h_{k'}$. By the definition of first segment, r can then lie on at most one such path $path(h_{k'}, r_{f'})$. By using the algorithm given in [8] which runs in time $O(\log n +$ size of the path constructed) it follows that the total time complexity is $O(n \log n)$ for all $O(n)$ shortest paths so constructed. □

To compute extremal segments in P for each r in $\mathbb{R}_k$, it suffices to compute the extremal segments inside the polygon S_k. To see this we observe the following: each polygon S_k is a subpolygon of P; thus if two points are visible in S_k they are visible in P. Furthermore, all vertices of S_k are vertices of P except for h_k.

We have established the correctness of the following lemma.

Lemma 5.5. *The extremal segment for each r in $\mathbb{R}_k$ (with respect to its relevant segment $s_k = (t_k, h_k)$) as computed in S_k is identical to that computed in P.*

Therefore the BSs for all reflex vertices contributing to the link center can be determined by the following algorithm in $O(n \log n)$ time. The correctness of the algorithm follows from the above.

Algorithm 5.3. Compute boundary segments
Input: The polygons S_k and the pairs $(s_k, \mathbb{R}_k)$
Output: The boundary segments for each reflex vertex contributing to the link center

```
1. for each subpolygon S_k do
2.    preprocess S_k for shooting toward s_k
3.    for each r in R_k do
4.       find extremal segment for (r, s_k) in S_k,
5.       determine the back-extension by shooting in P, and output it as
            a boundary segment for r.
```

5.3. *Computing the Relevant Boundary Segment (General Case)*

We have shown how to construct i-BSs given that any BS entering a region U is an $(i-1)$-BS. Now consider the case that for some convex vertex v one of its BSs

enters U "earlier," i.e., some k-BS intersects U for $k < i - 1$. The lemmas presented so far in this section are easily adapted to that situation.

Consider a particular region U of the window tree and assume that all children of its corresponding node have been processed (as in Section 4). Let m be the maximum index on BSs produced that intersect the region. Suppose first that all such BSs are m-BSs. Then we have the case considered above. Next assume that not all BSs are m-BSs. Clearly, all k-BSs for $k \leq m - 3$ are not relevant BSs. This holds since the covering radius of any such segment s' in U is three and thus U is contained in $P(s')$.

For those BSs s' which are $(m - 2)$-BSs we perform Algorithm 5.1 to obtain $(m - 1)$-BSs (if any). Then we take these and all other $(m - 1)$-BSs and apply Algorithm 5.1 again to obtain m-BSs. These are then treated together with the original m-BSs as discussed above.

Thus we get:

Theorem 5.6. *The BSs for all reflex vertices contributing to the link center can be determined in $O(n \log n)$ time.*

We have completed the main task for solving the problem of determining the link center, i.e., to identify the BSs contributing to the link center. For completeness we include, in Section 6, an algorithm for computing the link center from the BSs determined in Theorem 5.6.

6. Processing the Boundary Segments To Obtain the Boundary of the Link Center

The set of counterclockwise and the clockwise BSs of the reflex vertices computed in the previous sections contains all edges of the link center (some BSs are not collinear with any edges of the link center). To compute the link center from the BSs we apply the following algorithm.

Algorithm 6.1. Determining the boundary of the link center
Input: A simple polygon P, the polygon $V := VisPol(d, 2)$, a list of all reflex vertices in V, and their boundary segments.
Output: The link center of P
Let $\mathbb{R} = r_1, \ldots, r_{\text{last}}$ be the list of all reflex vertices of V (given in clockwise order).
Denote by $s_{i,1}$ and $s_{i,2}$ the counterclockwise and the clockwise boundary segments of r_i, respectively.
$LCB :=$ boundary of V {initialization of Link Center Boundary};
for $j := 1$ **to** 2 **do**
{$j = 1$ corresponds to a clockwise traversal and $j = 2$ corresponds to a counterclockwise traversal}
direct the edges of V and LCB according to c;

```
for i := 1 to last do
  if r_i is not removed then r := r_i;
  else {compute the first intersection point r of s_{i,j} and LCB}
    z_1 := delete[r_i];
    z_2 := the second endpoint of the most recently inserted segment
      in LCB;
    find the intersection point p of s_{i,j} with the boundary of V;
    if p ∈ CH_V(z_{3-j}, z_j) then skip the current i-step {s_{i,j} is not
      relevant for LCB}
    r := the intersection point of s_{i,j} with CH_LCB(z_{3-j}, z_j);
    {since CH_LCB(z_{3-j}, z_j) is convex, r can be located by binary search
      in log(n) time}
    e := the edge of LCB incident with r in direction c;
    remove e from LCB;
    delete[first endpoint of e] := r;
    repeat
      e := next edge of LCB in direction c;
      remove e from LCB;
      delete[first endpoint of e] := r;
      if s_{i,j} intersects e then
        compute the intersection point z of s_{i,j} and e;
        if shot_V(z, r) = segment(z, r) then
          {e is the edge from LCB hit by s_{i,j} (see proof of Lemma
            6.1 below)}
          insert segment(r, z) and segment(z, v(e)) in LCB,
          where v(e) is the endpoint of e that is in orientation c from
            z;
    until (a portion of) s_{i,j} has been inserted in LCB
```

Lemma 6.1. *Algorithm* 6.1 *computes the intersection of the regions determined by the BSs and its time complexity is* $O(n \log n)$.

Proof. Denote by B the boundary of V. Without loss of generality we consider the clockwise direction, i.e., $j = 1$. In the repeat-loop of Algorithm 6.1, the edges of B are examined in a clockwise direction until an edge e is discovered which intersects with some BS $s_{i,j}$ at a point z. Next the algorithm checks whether e is visible in P from r in direction $s_{i,j}$. For this, a shot toward r is made originating at point z. Assume that this shot reaches r. In this case, we claim that e is the edge from the link center boundary LCB "hit" by $s_{i,j}$. To prove this we need to show that before the ith iteration for $j = 1$, r_i is visible from e in V in the direction opposite to $s_{i,1}$ if and only if r_i is visible from e in the polygon V' determined by LCB. The following cases exist:

(i) The shot $s^* = shot_V(z, r_i)$ intersects an edge of both V and V'. Then r_i is not visible in V or in V'.

(ii) s^* intersects an edge f belonging to the set of edges of B-LCB (i.e., B without

LCB). Then r_i is not visible in V. Moreover, there exists a segment t of *LCB* inside V that cuts off a portion of B containing f. Then s^* cannot see r_i in V' because of (at least) t. Then r_i is not visible in V'.

(iii) s^* does not intersect an edge of B. Assume that s^* intersects an edge t of *LCB*. As e appears on *LCB* before t, the endpoints of t are between the second endpoint of e and r (clockwise). Therefore s^* hits another edge, edge w, of *LCB-B* (let t and w be the first two such edges). However, the angle between any two consecutive edges from *LCB-B* is convex. Thus there exist edges of *B-LCB* on $CH_V(w, t)$. It follows that s^* intersects some chain of B. The contradiction shows that s^* does not intersect *LCB* and thus r_i is visible both in V and in V'. The correctness of the iteration for $j = 2$ is proved in an analogous way.

Notice that we cannot directly shoot from r to *LCB* to locate the desired edge e, since *LCB* changes dynamically and such shooting cannot be performed efficiently. Instead, in Algorithm 4.5 we do the shooting in V polygon, which requires only $O(\log n)$ time after linear-time preprocessing of V.

We give an upper bound on the time complexity of Algorithm 6.1. Denote

n_1 = number of edges inserted in $LCB \leq 2*$(number of reflex vertices of V) $= O(n)$;

n_2 = number of edges removed from $LCB \leq$ (number of vertices of V) $+ n_1$ $= O(n)$;

n_3 = number of intersections computed $\leq n_1 + n_2 = O(n)$;

n_4 = number of shots produced $\leq$ number of reflex vertices of $V = O(n)$.

Then the time required by Algorithm 6.1 does not exceed

$$n_1 * O(1) + n_2 * O(1) + n_3 * O(1) + n_4 * O(\log n) = O(n \log n). \qquad \square$$

The results from this paper can be summarized in the following theorem.

Theorem 6.2. *For any simple n-vertex polygon P the link center and the link radius of P can be determined in $O(n \log n)$ time.*

7. Extensions

A central link segment in an n-vertex simple polygon P is a segment s of P that minimizes the number $\min_{q \in s} \max_{p \in P} \text{dist}(q, p)$, where $\text{dist}(q, p)$ is the link distance in P between two points p, q of P. Constructing the central link segment has applications in finding an optimal location of a robot in a polygonal region and in solving the problem of determining the minimum value k for which a given polygon is k-visible from some segment. The technique presented in this paper has been used in the design of an $O(n \log n)$-time algorithm for finding a central link segment in P; see [2]. (An alternate method is suggested in [9].)

We can generalize the problem of computing a link center of a simple polygon as follows. Let P be a simple polygon. Let Z be a set of points and let S be a set

of segments both located on the boundary of P. Let the sizes of S and Z be linear in the number n of vertices of P. Then the first algorithm described here can be adapted for finding the set of points in P which minimize the maximum link distance to all points in S and in P.

The algorithms presented here can also be used to solve the problem of determining the external link center of a polygon, i.e., that region(s) of the exterior of P for which each point inside the region minimizes the maximum (exterior) link distance to all points of P [3].

References

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
2. L. Alexandrov, H. Djidjev, and J.-R. Sack, Finding a central link segment of a simple polygon in $O(n \log n)$ time, Technical Report TR-89-7, Bulgarian Academy of Sciences, May 1989; also Technical Report SCS-TR-163, School of Computer Science, Carleton University.
3. L. Alexandrov, H. Djidjev, and J.-R. Sack, Finding the external link center of a simple polygon, unpublished manuscript, April 1990.
4. B. Chazelle, A theorem on polygon cutting with applications, *Proc. 23rd Annual IEEE Symp. on Foundations of Computer Science*, 1982, pp. 339–349.
5. B. Chazelle, Triangulating a simple polygon in linear time, *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science*, 1990, pp. 220–230.
6. H. Djidjev, Linear algorithms for graph separation problems, *Proc. SWAT '88*, Lecture Notes in Computer Science, Vol. 318, Springer-Verlag, Berlin, pp. 216–221.
7. H. N. Djidjev, A. Lingas, and J.-R. Sack, An $O(n \log n)$ algorithm for computing a link center in a simple polygon, *Proc. STACS '89*, Paderborn, February 1989, Lecture Notes in Computer Science, Vol. 349, Springer-Verlag, Berlin, pp. 96–107.
8. L. 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** (1987), 209–233.
9. Y. Ke, An efficient algorithm for link distance problems inside a simple polygon, *Proc. ACM Symp. on Computational Geometry*, Saarbrücken, June 1989, pp. 69–78.
10. W. Lenhart, R. Pollack, J.-R. Sack, R. Seidel, M. Sharir, S. Suri, G. Toussaint, S. Whitesides, and C. Yap, Computing the link center of a simple polygon, *Discrete and Computational Geometry* **3**(3) (1988), 281–293.
11. R. J. Lipton and R. E. Tarjan, A separator theorem for planar graphs, *SIAM Journal on Applied Mathematics* **36** (1979), 177–189.
12. R. Pollack, G. Rote, and M. Sharir, Computing the geodesic center of a simple polygon, *Discrete and Computational Geometry* **4**(6) (1989), 611–626.
13. J.-R. Sack, Movability of polygons in the plane, *Proc. STACS '85*, Saarbrücken, January 1985, Lecture Notes in Computer Science, Vol. 182, Springer-Verlag, Berlin, pp. 310–321; *Robotica* **5** (1987), 55–63.
14. J.-R. Sack, and S. Suri, An optimal algorithm for detecting weak visibility of a polygon, *IEEE Transactions on Computers* **39**(10) (1990), 1213–1219.
15. S. Suri, A linear time algorithm for minimum link paths inside a simple polygon, *Computer Vision, Graphics, and Image Processing* **35** (1986), 99–110.
16. S. Suri, Minimum link paths in polygons and related problems, Ph.D. Thesis, Department of Computer Science, Johns Hopkins University, August 1987.
17. G. T. Toussaint, Shortest path solves edge-to-edge visibility in a polygon, Technical Report SOCS-84.39, McGill University, Montreal, 1985.

Received September 26, 1990, and in revised form August 26, 1991.