2008－AL－116 (6)
2008／1／23

A new competitive strategy for exploring unknown polygons

Xuehou TAN
Tokai University, 317 Nishino, Numazu 410-0395, Japan
tan@wing.ncc.u-tokai.ac.jp

Abstract. We present a new, on-line strategy for a mobile robot to explore an unknown simple polygon, starting at a boundary point s, which outputs a so-called *watchman route* such that every interior point of P is visible from at least one point along the route. The length of the robot's route is guaranteed to be at most $4\sqrt{2}+1 \leq 6.7$ times that of the shortest watchman route that could be computed off-line. This gives a significant improvement upon the previous 26.5-competitive strategy, which was presented by Hoffmann et al. [?].
A novelty of our competitive strategy is a recursive procedure that reduces the polygon exploration problem to the subproblems of exploring two different types of reflex vertices. Moreover, our analysis of the competitive factor for a subproblem is based on the off-line $\sqrt{2}$-approximation algorithm for the watchman route problem [?] and a geometric structure called the *angle hull* [?].

1 Introduction

In the last decade, visibility-based problems of guarding, surveying or searching have received much attention in the commumities of computatonal geometry and on-line algorithms. Finding stational positions of guarding a polygonal region P is the well-known *art gallery problem*. The *watchman route problem* asks for a shortest route along which a mobile robot can see the whole region [?, ?, ?, ?, ?]. If the shape of the region P is not known to the robot in advance, it introduces the *on-line watchman route problem* or the *polygon exploration problem* [?, ?, ?, ?].

For the watchman route problem, an $O(n^4)$ time algorithm was first presented to solve the restricted version in which a starting boundary point s is given [?]. An $O(n^5)$ time algorithm was developed to remove the condition of a given starting point [?]. Recently, these results have been improved to $O(n^3 \log n)$ and $O(n^4 \log n)$, respectively [?]. On the other hand, a linear-time approximation solution to the watchman route problem with a given starting point s, which reports a watchman route guaranteed to be at most $\sqrt{2}$ times longer than the shortest watchman route through s, has been proposed in [?]. For the general problem without giving any starting point, the approximation factor is two [?].

In the polygon exploration problem, a starting point s on the boundary of P is given. A robot with a vision system that continuously provides the visibility of its current position walks to see the whole shape of P, starting from s. Once a corner of the polygon P is seen, it is memorized forever. When each point of P has at least once been visible, the robot returns to s. We are interested in a *competitive* exploration strategy that guarantees that the route of the robot will never exceed in length a constant times the length of the shortest watchman route through s. For the problem of exploring unknown rectilinear polygons, a $\sqrt{2}$-competitive strategy has been presented [?]. For simple polygons, Deng et al. were the first to claim that a competitive strategy does exist, but the constant is estimated to be in the thousands [?]. A factor of 133 was later given by Hoffmann et al. [?], which has recently been improved to $18\sqrt{2}+1 \leq 26.5$ [?].

In this paper, we present a new strategy for a mobile robot to explore an unknown simple polygon. First, we show that the polygon exploration problem can be reduced to the subproblems of exploring two different types of reflex vertices. For each subproblem, the off-line $\sqrt{2}$-approximation algorithm for the watchman route problem [?] and a geometric structure called the *angle hull* [?] are then used, so as to obtain a better competitive factor. With these ideas, we are able to prove that an unknown polygon can be explored by a route of length at most 6.7 times that of the shortest watchman route through s. This gives a significant improvement upon the previous 26.5-competitive strategy [?].

2 Preliminary

Let P be a simple polygon and s a point on the boundary of P. A vertex is *reflex* if its internal angle is strictly larger than π; otherwise, it is *convex*. The *shortest path tree* of s consists of all shortest paths from s to the vertices of P. The vertices touching a shortest path from right are called the *right reflex vertices*, or shortly, *right vertices*. The *left reflex vertices* or *left vertices* can be defined accordingly.

The polygon P can be partitioned into two pieces by a "cut" C that starts at a reflex vertex v and extends an edge incident to v until it first hits the polygon boundary. The piece of P containing s and including C itself is called the *essential piece* of C. We denote by $P(C)$ the essential piece of the cut C, and call v the *defining vertex* of C. See Fig. 1(a). A cut C_j *dominates* C_i if $P(C_j)$ contains $P(C_i)$ (Fig. 1(b)). We also say a point p *dominates* the cut C if p is not contained in $P(C)$. A cut is called the *essential cut* if it is not dominated by any other cuts. The watchman route problem is then reduced to that of finding the shortest route intersecting or visiting all essential cuts.

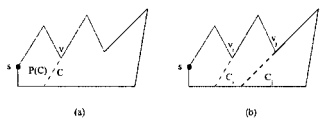

Figure 1: Essential cuts.

For short presentation, we denote by W_{opt} the shortest watchman route through s, and W_{app} the watchman route which is computed by the $\sqrt{2}$-approximation algorithm [7]. For a route R inside P, we denote by $|R|$ the length of R.

In the following, we briefly review the off-line $\sqrt{2}$-approximation algorithm [7], and then give the definition of angle hulls.

2.1 The $\sqrt{2}$-approximation algorithm

The reflection principle is used in most of the watchman route algorithms [3, 7, 9]. Let a and b denote the two points on the same side of a line L. Then, the shortest path visiting a, L and b in this order, denoted by $S(a, L, b)$, follows the reflection principle. That is, the incoming angle of $S(a, L, b)$ with L is equal to the outgoing angle of $S(a, L, b)$ with L. The reflection point on L can be computed by reflecting b across L to get its image b', and then reporting the intersection point of L with $\overline{ab'}$. See Fig. 2(a). Let $L(a)$ denote the point of L closest to a. The path consisting of $\overline{aL(a)}$ and $\overline{L(a)b}$, denoted by $S'(a, L, b)$, gives a $\sqrt{2}$-approximation of the path $S(a, L, b)$, since the angle $\angle a\ L(a)\ b'$ is at least $\pi/2$ (Fig. 2(a)). The same result also holds for a line segment l. See Fig. 2(b).

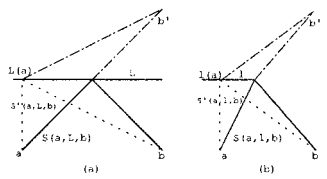

Figure 2: Approximating the reflection principle.

The idea of the $\sqrt{2}$-approximation algorithm is to repeatedly apply the approximation scheme designed for the reflection principle to essential cuts [7]. Let C_1, C_2, $\cdots$, C_m be the sequence of essential cuts indexed in clockwise order of their left endpoints, as viewed from s. Let $s = s_0 = s_{m+1}$. Given a point p in the polygon $P(C)$, we define the *image of p on the cut C* as the point of C that is closest to p inside $P(C)$.

Beginning with the starting point s, we first compute the images of s_0 on the cuts in the polygon P (or $P(C_0)$ [7]). Let s_1 denote the image of s_0 on C_1, s_2 the image of s_0 on C_2 and so on. The computation of s_0's images is terminated when the image s_{i+1} does not dominate the cuts $C_1, C_2, \ldots, C_i$ before it (Fig. 3). Then, we select a *critical* image from $s_1, s_2, \ldots, s_i$ as follows. If there exists an image s_h $(h < i)$ such that the image of s_h on C_{i+1}, which is computed in $P(C_h)$, dominates $C_{h+1}, \ldots, C_i$, we take the image s_h (e.g., the image s_1 in Fig. 3(a)) as the critical image. Otherwise, we take s_i (e.g., the image s_2 in Fig. 3(b)) as the critical image. Let s_k denote the chosen critical image. The images of s_k on the following cuts as well as the next critical image in the polygon $P(C_k)$ can similarly be computed [7]. This procedure is repeatedly performed until the image s_m on C_m is computed. See Fig. 3.

Let W_{app} denote the route which is the concatenation of the shortest paths between every

pair of adjacent critical images (including s_0 and s_{m+1}). Clearly, W_{app} is a watchman route (Fig. 3). An important property of W_{app} is that the reflection points (i.e., critical images) of W_{app} are guaranteed to be to the left of those of the shortest watchman route through s [7].

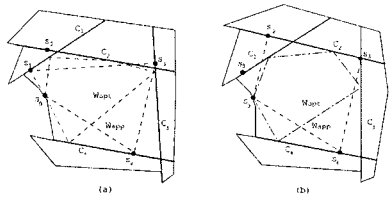

Figure 3: Critical images and routes W_{opt}, W_{app}.

Lemma 1 *(Tan [7]) For any instance of the watchman route problem with a given starting point s, $|W_{app}| \leq \sqrt{2}|W_{opt}|$ holds.*

2.2 Angle hulls

In an unknown polygon, exploring a reflex vertex v requires a little care. Since we do not know the cut defined by v, the point on the cut closest to the current position of the robot, say, p, cannot simply be found. This difficulty is overcome by using the circle spanned by v and by p [5]. Clearly, the intersection point of the circular arc with the cut is the point on the cut closest to p. This property leads to a study of angle hulls [5].

Let D denote a convex region in the plane. Suppose that a photographer follows a path to take a picture of D that shows as large a portion of D as possible but no white space or other objects, using a fixed angle lens, say, of 90°. All points enclosed by the photographer's path, and no other, can see two points of D at the right angle; we call this point set the *angle hull* of D, and denote it by AH(D). See Fig. 4(a).

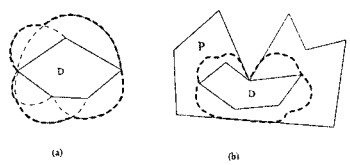

Figure 4: Angle hulls.

For the polygon exploration problem, the region D is defined as a *relative convex polygon* in P. That is, the shortest path between any two points of D inside P has to be contained in D. The photographer does not want any edges of P to appear in pictures; thus, the photographer's path may touch a vertex of P or overlap with a portion of the polygon edge. See Fig. 4(b).

In the outdoor setting, the perimeter of the angle hull is at most $\pi/2$ times the perimeter of D. In the indoor setting where D is contained in a simple polygon whose edges give rise to visibility constraints, we have the following result.

Lemma 2 *[5] Suppose that P is a simple polygon, and D is a relatively convex polygon (chain) inside P. The length of the perimeter of the angle hull AH(D), with respect to P, is less than 2 times the length of D's boundary.*

3 The 6.7-competitive strategy

We will present our 6.7-competitive strategy in a top-down manner. First, an overview of the competitive strategy is given. The details of the strategy are then described, and finally, the performance analysis is presented.

For simple presentation, we impose an ordering on the boundary points of P by a clockwise scan of the boundary, starting at s. So when we say a boundary point u is "smaller" (resp. "larger") than the other point v, it implies that u is encountered before (resp. after) v by a clockwise walker on the boundary, starting at s.

We say a vertex is *discovered* if it has ever been visible once from the robot. A left or right reflex vertex is *unexplored* as long as its cut has not been reached, and *fully explored* thereafter.

3.1 An overview of the strategy

First, the robot makes a clockwise tour to explore the right vertices, as many as possible, without considering to explore any left vertex of P. Next, the robot makes the other counterclockwise tour to explore the left vertices, as many as possible. During this counterclockwise tour, some of the right vertices having not yet been explored may become visible from the robot, as the left vertices that obstruct them from being visible from the first tour have been fully explored; those left vertices are taken as the starting points

for exploring the remaining right vertices. So the procedure for exploring right vertices is called again several times. In this way, all right and left vertices can eventually be explored.

Let *P-Exploration* denote the procedure for exploring a simple polygon. It mainly consists of a recursive procedure, which is named as P_r-*ExplorationRec*. The procedure P_r-*ExplorationRec* first explores the right vertices, as many as possible, and then calls the other recursive procedure for exploring the left vertices. The procedure for exploring the left vertices, denoted by P_l-*Exploration-Rec*, slightly differs from P_r-*ExplorationRec* because several further calls of P_r-*ExplorationRec* may be made within it.

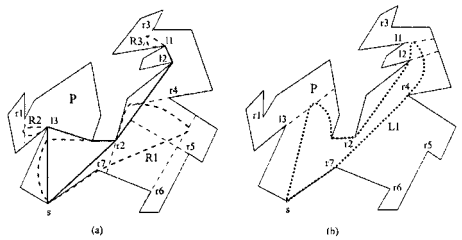

Figure 5: Exploring an unknown polygon.

Before describing *P-Exploration*, we give more definitions. Let CP denote the current position of the robot, which is initially set to s. Let *RightTarget* (resp. *LeftTarget*) denote the list of the right (resp. left) vertices to be explored, which have at least once been visible from the robot but have not yet been fully explored. Observe that the list *RightTarget* (resp. *LeftTarget*) dynamically changes when the robot walks to explore a right (resp. left) vertex.

Procedure *P-Exploration*(**in** P, **in** s)

1. Set *RightTarget* to the list of the right vertices, which are visible from s and ordered in clockwise order.
2. Call P_r-*ExplorationRec*(*RightTarget*, s).
3. The robot returns to s along the shortest path from CP, with turning points at polygon vertices.

Fig. 5 shows an example for exploring an unknown polygon. The routes $R1$ and $L1$ represent the first two routes for exploring right vertices and for exploring left vertices, and $R2$, $R3$ represent the two routes for exploring right vertices, which are discovered when the robot walks along the route $L1$. The route drawn in bold line in Fig. 5(a) shows the connection among the starting points for the second and higher level calls of P_r-*ExplorationRec* (or P_l-*Exploration-Rec*).

3.2 Exploring right vertices

In this section, we present a competitive strategy for exploring the right vertices. An intuition of our exploration strategy is to explicitly compute all critical images described in Section 2.1, with respect to the starting point s_r and the cuts having been explored by now. It is worth to pointing out that some of critical images may not be visited by the robot, although their positions are known to the robot. Note also that whether or not a cut is essential can be determined after its defining vertex is fully explored.

Denote by r the head of the list *RightTarget*, which is the target vertex that the robot is going to approach. Clearly, the value of r changes as soon as a smaller right vertex becomes visible from the robot. Denote by CI the current critical image, whose initial value is the starting point s_r. Also, denote by C the currently reached cut, and LI the (latest) image of CI on C. (The initial value of LI is CI.) In the approach to exploring the right vertex r, the variable CP changes, but the value of LI or CI does not. Again, as pointed out above, the position of CI or LI may not be reached by the robot.

In order to explore the vertex r, we make use of the following two circles. Denote by $Cir(CI)$ (resp. $Cir(LI)$) the clockwise oriented circle spanned by r and by the last vertex on the shortest path from CI to CP (resp. from LI to r). Note that the last vertex on the shortest path from CI to CP or from LI to r changes as soon as the corresponding path is changed.

To explore the very first vertex r, starting from s_r, the robot repeatedly walks on $Cir(CI)$. It may happen that the view to the target vertex r gets blocked (or when the boundary is hit). In this case, the robot walks straight toward the blocking vertex (or follows the boundary) until $Cir(CI)$ is encountered again. For an example, see the part of the robot's route from s_r to a

shown in Fig. 6(a).

Suppose below that C is the cut having just been explored, and thus $LI \neq CI$. Consider how to explore the second and the following right vertices, starting from s_r. Generally, the robot walks along C or the shortest path toward the vertex that blocks the view of r until the cut of r is reached, the part of $Cir(LI)$ contained in $P(C)$ or the part of $Cir(CI)$ outside of $P(C)$ is encountered. When the robot reaches (along C) the intersection point of C with the cut of r, the vertex r is fully explored. Whenever the part of $Cir(LI)$ contained in $P(C)$ or the part of $Cir(CI)$ outside of $P(C)$ is encountered, the robot changes to follow the encountered circle. In the former case, r is defintely explored. In the latter case, either the robot repeatedly walks on the encountered circles $Cir(CI)$ to explore r, or the robot returns to C again after the part of $Cir(CI)$ outside of $P(C)$ is walked through. After an *essential cut* is reached, we compute the new critical image, as what is done in [7]. This can be done as all the cuts between CI and CP are known to the robot. As soon as a new critical image is found, the variable CI is renewed. Moreove, the variable LI with respect to the new point CI is also maintained.

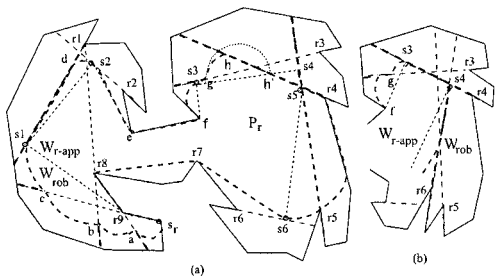

Figure 6: Exploring right vertices.

It may happen that the image of the new point CI on the cut having just been explored is not reached by the robot. This occurs when the robot reaches the cut of r at the intersection point of the cut of r with the previous cut C. For the example shown in Fig. 6(a), the point $s3$ on the cut of $r3$ is recognized as a new critical image when the robot reaches the intersection point of the cuts of $r4$ and $r5$, but CP is not equal to the image of $s3$ on the cut of $r4$. In this case, we compute this image of CI and let it be a critical image. In the following exploration, we take CP as a new starting point, like the original point s_r. See Fig. 6(a) for an example, where $s4$ is considered as a critical image and $s5$ is taken as a new starting point. Hence, two critical images on the same cut are reported in this case, which slightly differs from the $\sqrt{2}$-approximation algorithm [7]. (Note that for the example shown in Fig. 6(b), the point $s3$ on the cut of $r4$ is chosen as a critical image [7].)

As our strategy explores the right vertices as many as possible, it may thus happen that the robot loses sight of the next (discovered) right vertex, after the current target r is fully explored. In this case, the robot walks along the shortest path toward the head of *RightTarget* until it becomes visible again. For the example shown in Fig. 6(a), after $r2$ is fully explored, the robot moves along the shortest path to the point e, where $r4$ becomes visible again. The robot further moves along the polygon boundary to explore $r4$, and at f, the vertex $r3$ becomes visible and the target vertex then changes to $r3$.

It may also happen that the robot crosses the cut of a right vertex different from r. In this case, the former vertex (which is larger than r) is removed from *RightTarget*. After r is explored, it is deleted from *RightTarget*, too.

In the following, we first give a non-recursive procedure, denoted by P_r-*Exploration*, for exploring only the right vertices.

Procedure P_r-*Exploration*(**in** *RightTarget*, **in** s_r)

1. Set $CI \leftarrow s_r$.
2. **while** the list *RightTarget* is not empty **do**
   (a) The current *target* vertex, i.e., the vertex whose cut we are intending to reach at the moment, is always set to the head r of the list *RightTarget*. When no right vertices are visible from CP, the robot walks clockwise on the shortest path until the head of *RightTarget* becomes visible again.

   (b) If the very first right vertex, starting from CI, has not yet been fully explored, the robot repeatedly moves along the circles $Cir(CI)$ to explore it. Assume below that C is the cut having just been explored.

   (c) To explore the vertex r, the robot walks

along C or the shortest path toward the vertex blocking the view of r until the part of $Cir(LI)$ contained in $P(C)$ or the part of $Cir(CI)$ outside of $P(C)$ is encountered, or the cut of r is reached. When the specified part of $Cir(CI)$ or $Cir(LI)$ is encountered, the robot moves on that part to explore r. Also, the robot may return to C after the part of $Cir(CI)$ outside of $P(C)$ is walked through. After an essential cut is reached, the variable CI as well as LI is maintained. If the image of CI on the cut having just been explored is not reached by the robot, take CP a new starting point and then continue the exploration again.

When the robot walks along $Cir(CI)$ or $Cir(LI)$, its view to the target vertex may get blocked (or when the boundary is hit). In this case, the robot walks straight toward the blocking vertex (or follows the boundary) until $Cir(CI)$ or $Cir(LI)$ is encountered again. Also, the list *RightTarget* is maintained during the robot's walk.

3. The robot returns to the starting point s_r along the shortest path from CP.

Let us explain a little more on Step 2 of P_r-*Exploration*. Step 2(a) specifies which vertex or which cut we are intending to explore, and how to recovery sight of the target vertex. Step 2(b) devotes to the motion of the robot for exploring the very first right vertex from a starting point. Step 2(c) gives the method to approach the cut of the target vertex. See Fig. 6(a) for an example.

Lemma 3 *Let P_r denote a polygon, with a point boundary point s_r, such that the essential cuts of P_r are all defined by the right reflex vertices. A call of P_r-Exploration (***in** *RightTarget,* **in** *s_r) then explores the whole polygon P_r by reporting a route of length at most $2\sqrt{2}$ times the length of the shortest watchman route through s_r.*

Proof. To simplify the proof, we modify P_r as follows. For every (maximal) internal line segment through two polygon vertices on which the robot changes the circle to follow, we add two isometric edges to P_r by extending the line segment from its left endpoint until the extension is blocked by the route of the robot. See Fig. 6(a) for an example, where the introduced edges are shown in fat dashed line. The resulting polygon, denoted by P'_r, is still a simple polygon.

Let W_{rob} denote the route of the robot. The route W_{rob} is a relatively convex polygon inside P'_r, except for the situations in which W_{rob} makes left turns at some *interior* points of P'_r. It occurs when a smaller right vertex becomes visible. For the example shown in Fig. 7(a), in the approach to explore $r3$, the smaller vertex $r1$ becomes visible at the point x. Since the vertex y may block the view of $r1$, the robot further moves straight toward y. We call these interior points of P, the *pseudo-images*. So W_{rob} consists of the relatively convex chains, with the pseudo-images and the critical images as their endpoints.

Denote by T the list of all the pseudo-images and the critical images reported by calling P_r-*Exploration* (**in** *RightTarget*, **in** s_r), in clockwise order. Denote by W_{r-app} the route consisting of the shortest paths that connect every pair of two consecutive points of T (see Fig. 6(a)). We first claim that the route W_{rob} is of length at most the perimeter of the angle hull of W_{r-app} in P'_r. Assume that W_{rob} is a relatively convex polygon; otherwise, each relatively convex chain of W_{rob} is considered. The route W_{rob} is the same as the angle hull of W_{r-app}, except for the following two situations. The first exception is a trivial case, in which some parts of W_{rob} are the shortest paths from a critical image or a pseudo-image toward the vertex r or back to s_r, or even they enclose those of W_{rob} where a new starting point is reset (e.g., two segments $\overline{h's4}$ and $\overline{s4\ s5}$ of W_{r-app} enclose the segment $\overline{h's5}$ of W_{rob} in Fig. 6(a)). The second exception is that some circular parts of W_{rob} may be contained in the angle hull of W_{r-app}. For an example, the part of W_{rob} from g to h in Fig. 6(a) follows the circle spanned by f and $r4$, which is contained in the angle hull of the line segment $\overline{s3\ h'}$, where h' is the intersection point of $\overline{s3\ s4}$ with the cut of $r4$. The containment of W_{rob} in the angle hull of W_{r-app} comes from the fact that the semicircle on which the robot walks always contains the corresponding segment of W_{r-app}. Hence, our claim follows, and thus $|W_{rob}| \leq 2|W_{r-app}|$ holds.

Finally, we apply Lemma 1 to show $|W_{r-app}| \leq \sqrt{2}|W_{opt}|$. Suppose first that a pseudo-image x is used as the common endpoint of two shor-

est paths of W_{r-app}. It is easy to see that the (smaller) angle at x formed by two shortest paths of W_{r-app} is at least $\pi/2$ (Fig. 7(a)). Observe also that the portion of W_{opt} enclosed by these two shortest paths is a convex chain. See Fig. 7(a). Comparing with W_{opt}, the competitive factor for W_{r-app} to make a left turn at the pseudo-image x is no more than $\sqrt{2}$.

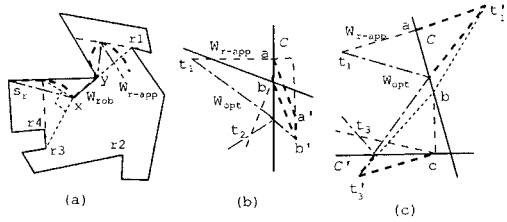

Figure 7: Illustraion for the proof of Lemma 3.

Assume below that W_{r-app} consists of the shortest paths that have only the critical images as their endpoints. Then, the route W_{r-app} is the same as W_{app}, except that two critical images on the same cut are reported by P_r-*Exploration*. In the following, we show that $|W_{r-app}| \leq \sqrt{2}|W_{opt}|$ also holds in this case. Let a and b denote two critical images in clockwise order, and $\mathcal{C}$ the cut on which a and b are. Recall that a is the image of the previous point CI on $\mathcal{C}$ by that momemt, and b is taken as a new starting point from then. Hence, the reflection point of W_{opt} is still to the right of a (if it reflects on $\mathcal{C}$). Since b is to the right of a on $\mathcal{C}$, the same reflection property still holds for the images after b, except for the point b itself. Thus, we only need to consider the parts of two routes before and after $\mathcal{C}$. If the reflection point of W_{opt} on the cut $\mathcal{C}$ is not between a and b, then the part of the route W_{r-app} between two points t_1 and t_2 is of length at most $\sqrt{2}$ times that part of the route W_{opt}, where t_1 and t_2 denote two intersection points of the routes W_{opt} and W_{r-app} before and after $\mathcal{C}$. This is because the route of W_{r-app} between t_1 and t_2 can be stretched into a convex chain that is enclosed in an obtuse-angled triangle with the longest edge $\overline{t_1 b'}$, where b' is the point obtained by reflecting t_2 across the cut $\mathcal{C}$. See Fig. 7(b) for an example, where $aa'b'b$ denotes a parallelogram. (We have assumed that W_{opt} makes a perfect reflection on $\mathcal{C}$ and that W_{rob} makes a right turn on $\mathcal{C}$ by at least $\pi/2$ [7].) If the reflection point of W_{opt} on $\mathcal{C}$ is between a and b, the proof can similarly be given. See also Fig. 7(c) for an example. In conclusion, we have $|W_{r-app}| \leq \sqrt{2}|W_{opt}|$. It completes the proof. □

There is a symmetric procedure P_l-*Exploration* for exploring the left vertices, which is identical to P_r-*Exploration*, except that left/right and clockwise/counterclockwise are exchanged.

We can now give the procedure P_r-*ExplorationRec*.

Procedure P_r-*ExplorationRec*(**in** *RightTarget*, **in** s_r)

1. Call P_r-*Exploration*(*RightTarget*, s_r).
2. Sort in counterclockwise order all left vertices, which are visible from the route of the robot produced by calling the procedure P_r-*Exploration*, and then, set *LeftTarget* to the list of these vertices.
3. Call P_l-*Exploration-Rec*(*LeftTarget*, s_l) by setting $s_l \leftarrow s_r$.

3.3 Exploring left vertices

As described in Section 3.1, the procedure P_l-*Exploration-Rec* first explores the left vertices of P, as many as possible, and then calls the procedure P_r-*ExplorationRec* several times, so as to further explore the right vertices that have at least once been visible from the robot but have not yet been fully explored. To this end, we maintain a list, say, *StartPoints*, to hold the *left* vertices l such that at least one right vertex becomes visible, for the *first* time, from the robot after the left vertex l is fully explored.

It is clear that some of the left vertices in *StartPoints* are descendants of others, and they have to be removed from the list *StartPoints*. Only maximal (highest up in the shortest path tree of s) are retained, which are the starting points for exploring the remaining right vertices.

Procedure P_l-*Exploration-Rec*(**in** *LeftTarget*, **in** s_l)

1. Call P_l-Exploration(*LeftTarget*, s_l).
2. Set *StartPoints* to the list of the left vertices in clockwise order such that some right vertices become visible, for the *first* time, after these left vertices are fully explored.

3. Clean up the list *StartPoints* so as to retain only those left vertices which are highest up in the shortest path tree of s.

4. **for** each vertex s_r of *StartPoints* **do**

   (a) Walk on the shortest path to s_r, and let *RightTarget* be the list of the right vertices in clockwise order, which are visible from s_r and have not been fully explored.

   (b) Call P_r-*ExplorationRec*(*RightTarget*, s_r).

In the following, we show that these local starting points have to be visited at least once by the shortest watchman route through s.

Lemma 4 *All the local starting points s_r for calling P_r-ExplorationRec have to be visited at least once by the shortest watchman route through s.*

Prof. Omitted in this extended abstract. □

3.4 Performance analysis

It is clear that the starting points for calling P_r-*Exploration* (P_l-*Exploration*) at the kth ($k \geq 1$) level differ from those at the $k+1$st level. Moreover, since the procedure P_r-*Exploration* (resp. P_l-*Exploration*) explores the right (resp. left) vertices as many as possible, any two routes of the robot output by calling the procedure P_r-*Exploration* (resp. P_l-*Exploration*) are mutually invisible (see also Fig. 5).

Lemma 5 *Any two routes of the robot output by calling the procedure P_r-Exploration (resp. P_l-Exploration) are mutually invisible, with a possible exception of their starting points.*

By now, we can obtain the main result of this paper.

Theorem 1 *For a polygon P and a starting point s on the boundary of P, a call of P-Exploration(P, s) explores P, which outputs a watchman route of length at most $4\sqrt{2}+1 \leq 6.7$ times the length of the shortest watchman route through s.*

Proof. It follows from Lemma 3 and Lemma 5 that all the routes W_{r-app} for exploring the right vertices (resp. left vertices), which are obtained by calling P_r-*Exploration* (resp. P_l-*Exploration*), cannot exceed in length $\sqrt{2}|W_{opt}|$. Therefore, all the robot's routes together, which are output by calling P_r-*Exploration* and P_l-*Exploration*, cannot exceed in length $4\sqrt{2}|W_{opt}|$.

The remaining task is to bound the path length caused by the walks during the **for** loops of P_l-*Exploration-Rec*. As shown in [5], all those walks together make up for an additional path length of at most $|W_{opt}|$ (see also Fig. 5). It completes the proof. □

References

[1] X. Deng, T. Kameda and C. Papadimitriou, How to learn an unknown environment, *Proc. FOCS'1991* 298-303.

[2] X. Deng, T. Kameda and C. Papadimitriou, How to learn an unknown environment I: The rectilinear case, *J. ACM* **45** (1998) 215-245.

[3] M. Dror, A. Efrat, A. Lubiw and J. S. B. Mitchell, Touring a sequence of simple polygons, *Proc. STOC'2003*, 473-482.

[4] F. Hoffmann, C. Icking, R. Klein and K. Kriegel, A competitive strategy for learning a polygon, *Proc. SoDA'1997*, 71-82.

[5] F. Hoffmann, C. Icking, R. Klein and K. Kriegel, The polygon exploration problem, *SIAM J. Comput.* **31(2)** (2001) 577-600.

[6] X.Tan, Fast computation of shortest watchman routes in simple polygons, *Inform. Process. Lett.* **77** (2001) 27-33.

[7] X.Tan, Approximation algorithms for the watchman and zookeeper's problems, *Discrete Applied Math.* **136** (2004) 363-376.

[8] X.Tan, A linear-time 2-approximation algorithm for the watchman route problem for simple polygons, *Theoretical Computer Science* **384** (2007) 92-103.

[9] X.Tan, T.Hirata and Y.Inagaki, Corrigendum to an incremental algorithm for constructing shortest watchman routes, *Int. J. Comput. Geom. Appl.* **9** (1999) 319-323.