SIAM J. COMPUT. © 2001 Society for Industrial and Applied Mathematics
Vol. 31, No. 2, pp. 577–600

THE POLYGON EXPLORATION PROBLEM*

FRANK HOFFMANN†, CHRISTIAN ICKING‡, ROLF KLEIN§, AND KLAUS KRIEGEL†

Abstract. We present an on-line strategy that enables a mobile robot with vision to explore an unknown simple polygon. We prove that the resulting tour is less than 26.5 times as long as the shortest watchman tour that could be computed off-line.

Our analysis is doubly founded on a novel geometric structure called the *angle hull*. Let D be a connected region inside a simple polygon, P. We define the *angle hull* of D, $\mathcal{AH}(D)$, to be the set of all points in P that can see two points of D at a right angle. We show that the perimeter of $\mathcal{AH}(D)$ cannot exceed in length the perimeter of D by more than a factor of 2. This upper bound is tight.

Key words. angle hull, competitive strategy, computational geometry, curve length, motion planning, navigation, on-line algorithm, optimum watchman tour, polygon, robot

AMS subject classifications. 68U05, 68U30

PII. S0097539799348670

1. Introduction.

In the last decade, the path planning problem of autonomous mobile systems has received a lot of attention in the communities of robotics, computational geometry, and on-line algorithms; see, e.g., Rao et al. [23], Blum, Raghavan, and Schieber [5], and the surveys by Mitchell [21] in Sack and Urrutia [25] and by Berman [4] in Fiat and Woeginger [13]. We are interested in strategies that are correct, in that the robot will accomplish its mission whenever this is possible, and in performance guarantees that allow us to relate the robot's cost to the cost of an optimal off-line solution or to other complexity measures of the scene.

In this work we are addressing a basic problem in this area. Suppose a mobile robot has to explore an unknown environment modeled by a simple polygon. The robot starts from a given point, s, on the polygon's boundary. It is equipped with a vision system that continuously provides the visibility of the robot's current position. When each point of the polygon has at least once been visible, the robot returns to s.[1]

In the on-line polygon exploration problem we ask for a *competitive* exploration strategy that guarantees that the robot's path will never exceed in length a constant *competitive factor* times the length of the optimum watchman tour through s, i. e., of the shortest tour inside the polygon that contains s and has the property that each point of the polygon is visible from some point of the tour. This approach to evaluating the performance of an on-line strategy goes back to Sleator and Tarjan [27]. A priori it is not clear whether a competitive exploration strategy exists.

*Received by the editors June 10, 1999; accepted for publication (in revised form) November 6, 2000; published electronically October 11, 2001. This work was supported by the Deutsche Forschungsgemeinschaft, grant Kl 655/8-3. This is the final version of conference papers that have appeared in the proceedings of SODA '97 and WAFR '98 (section 2; see [15, 16]) and SWAT '98 (section 3; see [17]).

http://www.siam.org/journals/sicomp/31-2/34867.html

†Freie Universität Berlin, Institut für Informatik, Takustrasse 9, D-14195 Berlin, Germany (hoffmann@inf.fu-berlin.de, kriegel@inf.fu-berlin.de).

‡FernUniversität Hagen, Praktische Informatik VI, Feithstrasse 142, D-58084 Hagen, Germany (icking@feu.de).

§Universität Bonn, Institut für Informatik I, Römerstrasse 164, D-53117 Bonn, Germany (rolf.klein@uni-bonn.de).

[1] In the absence of holes, the robot has seen each point inside the polygon as soon as it has seen each point on its boundary.

Even the *off-line* version of the polygon exploration problem is not easy. Here we are given a simple polygon and have to compute the optimum watchman tour through a specified boundary point, s. Initially this problem has been suspected to be NP-hard. Chin and Ntafos [9] were the first to provide a polynomial time solution. They have shown how to compute the optimum watchman tour in time $O(n^4)$, where n denotes the number of vertices of the polygon. Later, their result has been improved by Tan and Hirata [28]; see also the updates in [14, 29].

Carlsson, Jonsson, and Nilsson [7] have proven that the optimum watchman tour without a specified point s can be computed in time $O(n^3)$. Furthermore, Carlsson and Jonsson [6] proposed an $O(n^6)$ algorithm for computing the shortest path inside a simple polygon from which each point of the boundary is visible when start and end points are not specified. In these papers it is always assumed that the range of the robot's visibility is unbounded. Some authors have also studied the case of limited visibility, e.g., Arkin, Fekete, and Mitchell [3] and Ntafos [22].

As to the *on-line* version of the polygon exploration problem, Deng, Kameda, and Papadimitriou [11] were the first to claim that a competitive strategy does exist. In their seminal paper they discussed a factor of 2016 for a greedy off-line approach which has to be implemented as an on-line strategy. For the rectilinear case, they gave a complete and elegant proof in [12]; here the greedy strategy can be applied that performs surprisingly well.

The first proof for the more difficult case of nonrectilinear simple polygons has been given in our conference paper [15]. There we have provided an on-line exploration strategy and sketched a proof that the tour it generates in any polygon is not longer than 133 times the length of the optimum watchman tour. One of the main difficulties with this analysis was in establishing reasonably sharp length estimates for robot paths of complex structure and in relating them to the optimum watchman tour.

The present paper contains the first complete presentation and analysis of an exploration strategy for simple polygons. As compared to the conference version [15], this full paper has been greatly simplified and describes a new analysis that is built on an interesting geometric relation between the robot's path and the optimum watchman tour. This relation is expressed in terms of the *angle hull*, a novel geometric structure. With these improvements we are able to show that an unknown polygon can be explored, from a given boundary point, s, by a tour at most 26.5 times as long as the shortest watchman tour containing s.

Of course, there is still a considerable gap between this upper bound and the lower bound of $(1 + \sqrt{2})/2$ given in [11]; however, experiments with our new strategy suggest that its actual performance is much better than the bound proven here; we conjecture a number far below 10.

The organization of this paper is as follows. Section 2 contains a hierarchical description of the strategy and of its analysis. In section 2.1 we first discuss how to explore a single corner, that is, a single reflex vertex one of whose adjacent edges has not yet been visible. By *reflex vertex* we denote a vertex of the polygon whose internal angle exceeds 180°; all other vertices are called *convex*. The robot explores these corners in a sophisticated order: Of all reflex vertices that touch the visible area from the right, the robot attempts to explore the one that is clockwise first on the polygon's boundary as seen from the starting point, s. However, the vertex hereby specified may change as the robot moves. From our angle hull result we obtain a bound on the length of the resulting path in terms of the length of the shortest path that leads to the final position.

Then, in section 2.2, we use this technique for efficiently exploring groups of right reflex vertices in clockwise order. The length of the resulting local tour is shown to be bounded by the perimeter of the relative convex hull of certain *base points* times a constant. In analogy to the standard definition of convex hulls, the *relative convex hull*, $\mathcal{RCH}(D)$, of a subset D of a polygon P is the smallest subset of P that contains D and, for any two points of D, the shortest path in P connecting them.

In section 2.3 we show how the robot recursively detects and explores an exhaustive system of groups of right and left reflex vertices. Groups of vertices that are on sufficiently different recursive levels give rise to base point sets whose $\mathcal{RCH}$s are mutually invisible. Therefore, the sum of their hulls' perimeters is less than the perimeter of the $\mathcal{RCH}$ of their union. In Lemma 2.6 we will show that each base point can see two points of the optimum watchman tour, W_{opt}, at an angle of $90°$; therefore, all base points must be contained in the angle hull of W_{opt}. Consequently, the perimeter of the $\mathcal{RCH}$ of the base points must be less than the perimeter of the $\mathcal{RCH}$ of the angle hull of W_{opt}. The latter, in turn, can only be less than or equal to the perimeter of the $\mathcal{RCH}$ of W_{opt} itself, because the perimeter of the $\mathcal{RCH}$ of a connected object is no longer than the perimeter of the object itself. Now we can apply our angle hull result a second time in estimating the perimeter of the angle hull of W_{opt} against the length of W_{opt} itself. This way we have bounded the length of the whole exploration path walked by the robot by a multiple of the length of the optimum watchman path, W_{opt}.

Our analysis greatly benefits from this new geometric structure we propose to call the *angle hull*. Let D be a simple polygon contained in another simple polygon, P. Then the angle hull, $\mathcal{AH}(D)$, of D consists of all points in P that can see two points of D at an angle of $90°$. The boundary of $\mathcal{AH}(D)$ can be described as the path of a diligent photographer who uses a $90°$ angle lens and wants to take a picture of D that shows as large a portion of D as possible but no walls of P. Before taking the picture, the photographer walks around D, in order to inspect all possible viewpoints.

In section 3 we prove that the photographer's path is at most twice as long as the perimeter of her model, D.

2. The strategy and its analysis.

Let P be a simple polygon and let s be a point on its boundary. The *shortest path tree* of s consists of all shortest paths from s to the vertices of P. Its internal nodes are reflex vertices of P. Those vertices touching a shortest path from the right are called *right reflex vertices*; left reflex vertices are defined accordingly. If we follow the shortest path from s to reflex vertex v, one of its adjacent polygon edges remains invisible until v is actually reached. The extension into the polygon of this invisible edge is called a *cut* of P with respect to s.

Exploring a polygon P is equivalent to visiting all of its cuts with respect to the start point s.

Figure 2.1 shows an example of the *optimum watchman tour*, W_{opt}, containing a boundary point, s. Tan and Hirata [28] have provided an off-line algorithm for computing W_{opt} within time $O(n^2)$ for a polygon of n edges.

We say a vertex has been *discovered* after it has been visible at least once from the robot's current position. A reflex vertex is *unexplored* as long as its cut has not been reached, and *fully explored* thereafter.

In an unknown polygon, even exploring a single reflex vertex requires a little care. For example, one cannot afford to go straight to the vertex in order to get to its cut: The cut could be passing by the start point very closely, so that a much shorter path would be optimal.

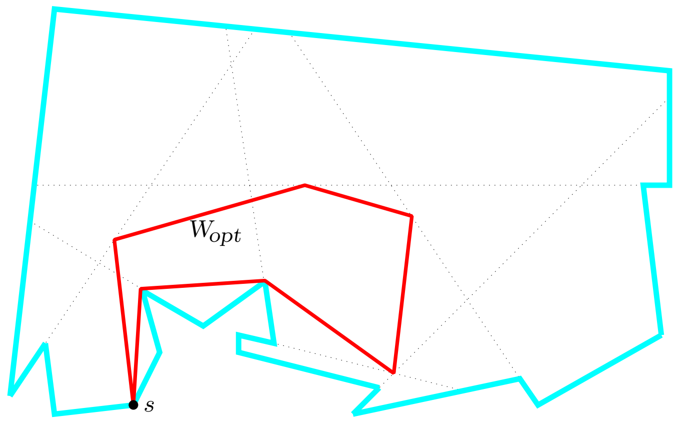

FIG. 2.1. *The optimum watchman tour visits all cuts of the polygon.*

We avoid this difficulty as follows. Whenever the robot wants to explore a right reflex vertex, r, visible from some local start point, p, it approaches r along the clockwise oriented circle *spanned* by p and by r, denoted by *circ* (p, r), i.e., the smallest circle that contains p and r.

Consequently, when the robot reaches the cut of r at some point c, the ratio of the length of the circular arc from p to c over their euclidean distance is bounded by $\frac{\pi}{2} \approx 1.57$.

One might wonder if the subproblem of exploring a single vertex can be solved more efficiently by using curves other than circular arcs. This is, in fact, the case; Icking, Klein, and Ma [20] have shown that an optimum ratio of ≈ 1.212 is achieved by curves that result from solving certain differential equations. However, these curves are lacking a useful property possessed by circular arcs: The intersection point, c, of the circular arc with the cut is just the point on the cut closest to p, due to Thales's theorem.[2] This property turns out to be very helpful in our analysis.

In rectilinear polygons, the cut of each visible reflex vertex is known, and two cuts can cross only perpendicularly. This makes it possible to apply a simple greedy exploration strategy: The robot always walks to the cut of the next reflex vertex, in clockwise order, one of whose edges is invisible; see Deng, Kameda, and Papadimitriou [12].

For general polygons, this greedy approach is bound to fail, as Figure 2.2 illustrates. The example polygon shown there suggests exploring left and right reflex vertices separately. However, it is not really obvious how to do this in general, since, e.g., the existence of a left reflex vertex at the end of a long chain of right vertices is initially not known to the robot. Therefore, it seems necessary to partition left and right reflex vertices into compact groups that can be explored one by one.

2.1. Exploring a single vertex. The essential subtask of the robot's strategy is in exploring a single vertex. This is handled by the following procedure, *ExploreRightVertex*. We first list the complete pseudocode of the procedure, then we explain its steps and notation in detail.

[2]Thales's theorem: an angle inscribed in a semicircle is always a right angle.

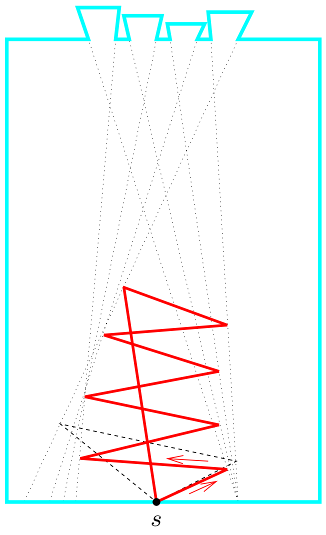

FIG. 2.2. *Visiting cuts in the order in which their vertices appear on the boundary does not lead to a competitive strategy.*

```
PROCEDURE ExploreRightVertex ( inout TargetList, inout ToDoList );
  BasePoint := CP;
  Target := First (TargetList);
  if Target not visible then
    walk on shortest path from BasePoint to Target
    until Target becomes visible;
  Back := last vertex before CP on shortest path from BasePoint to CP;
  walk clockwise along circ (Back, Target)
   while maintaining TargetList and ToDoList
    whenever First (TargetList) changes let Target := First (TargetList);
    whenever Back becomes invisible update Back;
    exceptions for walking along the circle:
      if the boundary of P blocks the walk on the current circle then
        walk clockwise along the boundary
        until the circular walk is again possible;
      if Target is becoming invisible then
        walk towards Target
        until the blocking vertex is reached;
  until Target is fully explored;
end ExploreRightVertex;
```

We are using the abbreviation *CP* to denote the robot's current position. Procedure *ExploreRightVertex* works on two lists of vertices, *TargetList* and *ToDoList*. On entry, *TargetList* contains a list of right vertices, sorted in clockwise order along the boundary, that have already been discovered but not yet explored. When *ExploreRightVertex* is called for the very first time, *TargetList* contains exactly those right vertices that are visible from the start point, s, and have an invisible edge. *ToDo-*

List can be thought of as a long-term agenda that is passed to *ExploreRightVertex*; however, this procedure will only add to this list but not carry out one of the tasks.

In our pseudocode, *CP* (current position) is a global variable whose value can be changed only by **walk** statements. The robot's current position on calling *Explore-RightVertex* is called a *base point*.

The robot wants to explore the first vertex, *Target*, of *TargetList*. This vertex may have been discovered at an earlier stage, so that it may no longer be visible from the current position. In this case, the robot walks along the shortest path towards *Target* until it becomes visible again. Note that this shortest path is known to the robot; in fact a shortest path is always known between two points that have been seen (which would not be the case for polygons with holes).

Now the robot starts approaching *Target* along the circular arc spanned by the base point and by *Target*. On the way, a new right vertex, r, may be discovered. If one of its edges is invisible, r gets inserted into *TargetList*, provided that a certain criterion is met. Namely, the shortest path from the current *stage point*—a vertex defined one level up in the strategy—to r must not contain left turns. A right vertex that violates this criterion is ignored for now. A precise definition of a stage point is given in section 2.2.

It may happen that the vertex r newly discovered and inserted into *TargetList* comes before *Target* in clockwise order. In this case the robot ceases approaching its old target and starts exploring, from its current position, vertex r. This way, the vertex *Target* currently under exploration may repeatedly change.

It may also happen that the robot loses sight of the base point from which the current execution of procedure *ExploreRightVertex* has started. Namely, the robot's view of the base point may become obstructed by some left or right reflex vertex, b_1; examples will be shown in Figure 2.3. In this case, the exploration of the current *Target* no longer proceeds along the circle spanned by the base point and by *Target*; instead, it switches to the circle spanned by *Target* and by b_1. As the robot continues, its view of b_1 may become obstructed by some reflex vertex b_2, and so on. These reflex vertices, b_1, b_2,…, b_i define the shortest path from the base point to the robot's current position, and the robot explores *Target* along the circle spanned by *Target* and b_i. This last reflex vertex of the chain is named *Back* in the code. If the robot reaches the cut of *Target* at some point c, it arrives there at a right angle by the Thales property. Consequently, the shortest path in P from the base point to the cut goes through b_1, b_2,…, b_i and ends in c.

If the robot crosses the cut of a right vertex different from *Target* the former vertex is removed from *TargetList* because it has been explored on the way, this is done in the "while maintaining *TargetList*" step. Eventually, the target itself is deleted from the list when its cut has been reached.

When a right reflex vertex is explored, all of its children in the shortest path tree have already been discovered. Those right vertices having a left child are inserted into *ToDoList*, as candidates for future stages, together with references to their left children.

Finally, there are some exceptional events procedure *ExploreRightVertex* needs to take care of. If the robot's circular exploration path hits the boundary of the polygon, the robot follows the boundary until a circular path again becomes possible. If the robot's view of the target vertex is about to be blocked, the robot walks straight to the blocking vertex and continues from there on a circular path.

For ease of reference we summarize the rules by which *ExploreRightVertex* proceeds.

1. The current *target* vertex, i. e., the vertex whose cut we are intending to reach at the moment, is always the clockwise first among those right reflex vertices that have been discovered but not yet fully explored, i. e., the first element of *TargetList*. Only such right vertices can be in *TargetList* whose shortest paths from the stage point make only right turns.
2. To explore a right reflex vertex, r, we follow the clockwise oriented circle spanned by r and by the last vertex before CP on the shortest path from the base point to CP.
3. When the view to the current target vertex gets blocked (or when the boundary is hit) we walk straight towards the blocking vertex (or follow the boundary) until motion according to rule 2 becomes possible again.

Figure Figure 2.3 demonstrates how this strategy works. Initially, r_3 is the only right vertex visible; consequently, *TargetList* contains only r_3, and the robot's path begins with a circular arc spanned by s and by r_3. At point a, right vertex r_2 becomes visible. It is situated before r_3 on the boundary; therefore, the robot switches to exploring r_2, according to rule 1. Note that the circle spanned by s and by r_2 is passing through a, too, so that it is in fact possible to apply rule 2 at this point.

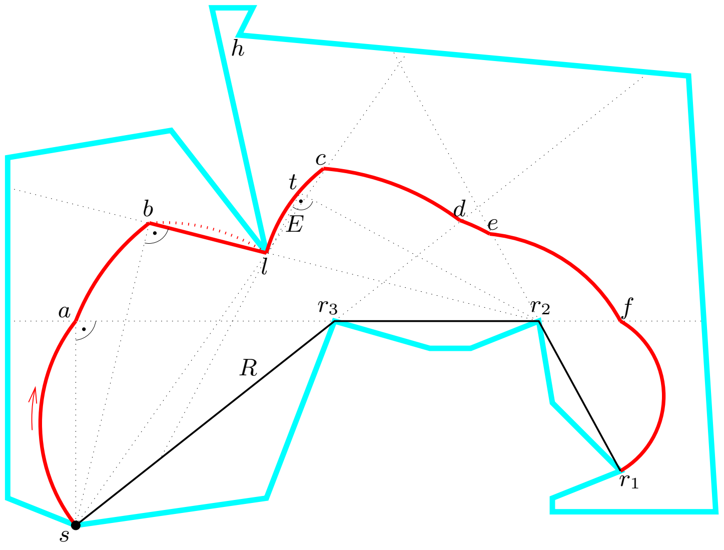

FIG. 2.3. *While executing ExploreRightVertex, the target vertex is initially r_3, then changes to r_2 and finally to r_1.*

At point b, vertex r_2 would become invisible if the robot were to follow the circular arc. But now rule 3 applies, causing the robot to walk straight to the left reflex vertex l. From there, a circular motion is again possible; but the shortest path from s to CP now contains vertex l. By rule 2, the robot continues its approach to r_2 along the arc spanned by l and by r_2.

Notice that at vertex l, also the right vertex h becomes visible, but it is ignored because its shortest path from s makes a left turn at l.

From c on, the shortest path to s is the line segment. Since the circle spanned by s and by r_2 is passing through c, the robot proceeds along that circle, applying rule 2.

At d, the shortest path to s changes again; now it contains vertex r_3. The robot walks along the circle spanned by r_3 and r_2, and gets to point e from which vertex r_1 becomes visible. From here, the robot explores r_1, following the circle spanned by r_3 and by r_1, the former changing to r_2 at f. Eventually, the robot arrives at r_1, thereby fully exploring r_1. Here procedure *ExploreRightVertex* terminates.

In order to provide an upper bound on the length of the resulting path we need to give the definition of the angle hull. A detailed discussion of this topic will follow in section 3.

Let D be a bounded, connected region in the plane. For convenience, we shall assume that D is a simple polygon, but our results can easily be generalized to curved objects by approximation. Now suppose that a photographer wants to take a picture of D that shows as large a portion of D as possible, but no white space. The photographer is using a fixed angle lens. For now, we assume that the angle equals 90°; later, at the end of section 3.2, we will see how to generalize to arbitrary angles.

Before taking the picture, the diligent photographer walks around D and inspects all possible viewpoints. We are interested in comparing the length of the photographer's path to the perimeter of the object, D.

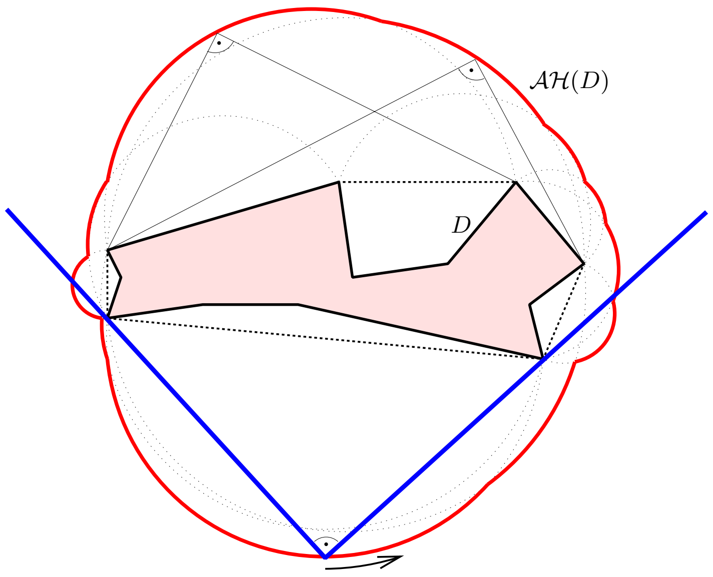

FIG. 2.4. *Drawing the angle hull $\mathcal{AH}(D)$ of a region D.*

In the simple outdoor setting there are no obstacles that can obstruct the photographer's view of D; this situation is depicted in Figure 2.4. At each point of the path, the two sides of the lens' angle touch the boundary of D from the outside, in general at a single vertex each.

While the right angle is touching two vertices, v and w, of D, its apex describes a circular arc spanned by v and w, as follows from Thales' theorem. All points enclosed by the photographer's path, and no other, can see two points of D at a 90° angle; we call this point set the *angle hull* of D and denote it by $\mathcal{AH}(D)$.

Only such vertices can be touched by the right angle that are situated on the convex hull of D. Consequently, the photographer's path depends on the convex hull, $\mathcal{CH}(D)$, of D, rather than on D itself, therefore we have $\mathcal{AH}(D) = \mathcal{AH}(\mathcal{CH}(D))$.

It is not hard to see that, without further obstacles, the perimeter of the angle hull is at most $\pi/2$ times the perimeter of D; the worst case occurs when D is a line segment or a rectangle; see [16].

However, for our application we need to analyze the indoor setting where D is contained in a simple polygon P whose edges give rise to visibility constraints. The photographer does not want any wall segments to appear in the picture; thus, the viewing angle can now be constrained in different ways: Either side may touch a convex vertex of D that is included in the angle, as before, or it may touch a reflex vertex of P that is excluded; see Figure 2.5.

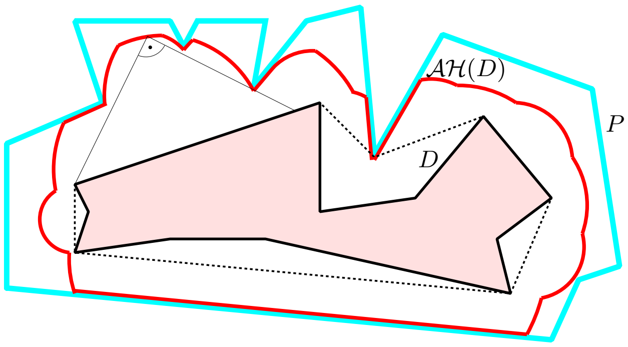

Fig. 2.5. *The angle hull $\mathcal{AH}(D)$ inside a polygon P.*

Any combination of these cases is possible.

As a consequence, the photographer's path contains circular arcs spanned by vertices of D and of P; in addition, it may contain segments of edges of P that prevent the photographer from stepping back far enough; see Figure 2.5.

Formally, we define the angle hull, $\mathcal{AH}(D)$, of D with respect to P to be the set of all points of P that can see two points of D at a right angle. Its boundary equals the photographer's path. In the indoor setting, the angle hull $\mathcal{AH}(D)$ depends only on the relative convex hull, $\mathcal{RCH}(D)$, of D; in other words $\mathcal{AH}(D) = \mathcal{AH}(\mathcal{RCH}(D))$.

In section 3 we show that the angle hull can have at most twice the perimeter of D.

Coming back to our exploration strategy, the crucial observation is that its path is essentially an angle hull (AH).

Lemma 2.1. *Assume the robot starts at the base point and invokes procedure ExploreRightVertex. Suppose this procedure terminates with the robot reaching the cut of target vertex r_1 at point c. Then the shortest path, R, inside P from the base point to the cut also reaches the cut at c. Moreover the robot's path is part of the boundary of the angle hull $\mathcal{AH}(R)$ of R except for straight line segments leading to blocking vertices.*

Proof. In the discussion of procedure *ExploreRightVertex* above in this section, we have already shown that the robot's path and the shortest path to the cut arrive at the same point c.

Now let t be a point on the robot's path that is not contained on a straight line segment of the path. Assume that, at t, the robot is exploring right reflex vertex r_2, as in the example shown in Figure 2.3. Since r_2 and r_1 are in convex position relative to the base point, vertex r_2 lies on the shortest path, R, from the base point to r_1.

Now consider the shortest path, T, from the base point to t. As a consequence of rule 2 and by Thales' theorem, the last line segment, E, of T is perpendicular to the line through t and r_2. Since the backward prolongation of E is bound to hit R, we know that point t can see two points of R at a right angle. Thus, t belongs to the angle hull $\mathcal{AH}(R)$; it lies on the boundary because the angle's sides are both touching reflex vertices of P or endpoints of the path R. $\square$

To estimate the length of the path from the base point to the cut we make use of the analysis of the angle hull from section 3.

LEMMA 2.2. *The robot's path from the base point to the cut of the target vertex explored by procedure* ExploreRightVertex *is not longer than twice the length of the shortest path.*

Proof. If the robot's path contains straight line segments leading to blocking vertices, like the segment from b to l in Figure 2.3, these segments are replaced by circular arcs in the angle hull $\mathcal{AH}(R)$. Thus, the robot's path to the cut of r_1 cannot be longer than the angle hull's perimeter. If it ends at the point r_1 itself, as in Figure 2.3, we can apply Theorem 3.5—which states that the arc length of the angle hull of a polygon D in P is less than twice as long as D's boundary—to the shortest path as D and obtain the desired upper bound.

If the robot reaches the cut of r_1 at some point different from r_1, we can arrive at the same conclusion using Corollary 3.6. $\square$

It is important to note that procedure *ExploreRightVertex* ignores such vertices as h in Figure 2.3, whose shortest paths from the current stage point include left turns. Otherwise, it would not be clear how to apply Lemma 2.1.

There is a symmetric procedure *ExploreLeftVertex* which is identical to *ExploreRightVertex*, except that left/right and clockwise/counterclockwise are exchanged.

2.2. Exploring a group of vertices.

Each exploration of a group of vertices starts from a *stage point*. The importance of stage points lies in the fact that they are visited by the optimum watchman tour, W_{opt}, too. The first stage point encountered is the robot's start point, s. All stage points are vertices of the shortest path tree of s; the shortest path from s to any vertex of a group leads through the group's stage point.

The exploration of a group of right vertices is performed by procedure *ExploreRightGroup*.

```
PROCEDURE ExploreRightGroup ( in TargetList, out ToDoList );
  StagePoint := CP;
  ToDoList := empty list;
  while TargetList is not empty do
    ExploreRightVertex (TargetList, ToDoList);
      (* CP is now on the cut, C, of the last target. *)
    walk to the point on C that is closest to StagePoint
      while maintaining TargetList and ToDoList;
  walk on the shortest path back to StagePoint;
end ExploreRightGroup;
```

The stage point of a right group is always a left vertex. Initially, *ToDoList* is empty, whereas *TargetList* contains a sorted list of unexplored right vertices whose shortest path from the base point makes only right turns. Among them are all unexplored right vertices visible from *StagePoint*.

Roughly, the group exploration proceeds by repeatedly calling procedure *Explore-RightVertex* introduced in section 2.1 until *TargetList* becomes empty. Afterwards, all right vertices initially present in *TargetList* have been explored, together with their *purely right descendants* in the shortest path tree of s. Here, vertex w is called a *purely right* descendant of vertex v in the shortest path tree of s if w is a right vertex and if the path from v to w makes only right turns. This set of vertices constitutes a *group*, by definition.

On returning from a call to *ExploreRightVertex* the robot has just explored the clockwise first vertex of *TargetList* and is now situated on this vertex's cut. Before it continues, the robot walks along this cut to the point closest to the stage point; this will be the base point in the next execution of *ExploreRightVertex*. The reason for this step will become clear in the proof of Lemma 2.5; essentially, it keeps the robot closer to the optimum watchman tour.

Once the last vertex of *TargetList* has been explored, the robot walks back to the stage point, thus completing the exploration of the group. Now *ToDoList* contains, of all right vertices explored, those who have left children, together with references to the latter.

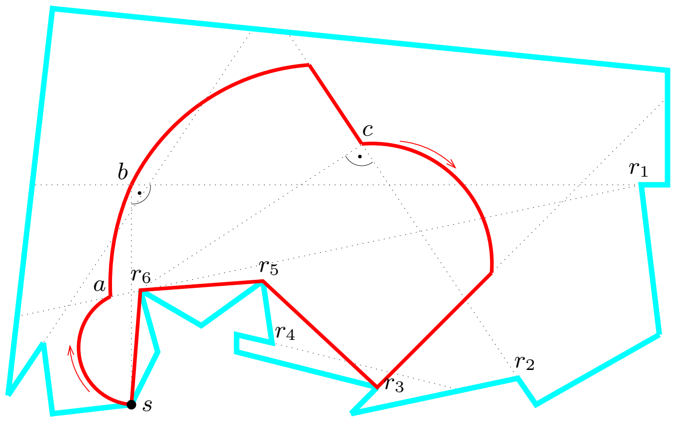

FIG. 2.6. *Exploring a group of right vertices.*

For an example, see Figure 2.6. Point s is the stage point and also the first base point, and *ExploreRightVertex* is called with *First* (*TargetList*) $= r_6$. While exploring r_6, point r_1 is discovered at point a and becomes *First* (*TargetList*). At $CP = b$ procedure *ExploreRightVertex* returns. Meanwhile, r_2 and r_5 have been added to *TargetList* while r_1 and r_6 have been removed. Point b is also the closest point to s on the current cut.

As we continue with exploring *First* (*TargetList*) $= r_2$, point r_5 gets explored on the way. Once the cut of r_2 is reached, we walk to c, the closest point to s on the cut. Similar for r_3; while walking along the cut to the point closest to s, which is r_3 itself, r_4 gets explored and no unexplored right vertices remain.

As before with *ExploreRightVertex*, for *ExploreRightGroup* we also have a symmetric counterpart, *ExploreLeftGroup*.

First we prove a useful structural result which says that the shortest paths to the base points fan out in the same order as the base points are generated.

LEMMA 2.3. *Suppose that procedure ExploreRightGroup generates the base points $b_1, \ldots, b_m$ in m consecutive calls of subroutine ExploreRightVertex. Then the* *shortest paths from the stage point to* $b_1, \ldots, b_m$ *are in clockwise order.*

Proof. Let base point b_i be situated on the cut of right reflex vertex v_i. Since each call to *ExploreRightVertex* explores the clockwise first right vertex that is still unexplored, $v_1, \ldots, v_m$ appear in clockwise order on the boundary. The stage point must be situated below the cuts of v_i and v_{i+1} as these are unexplored right vertices. The same holds for the last point, p, the shortest paths from the stage point to b_i and b_{i+1} have in common. Moreover, b_i must be below the cut of v_{i+1} because the latter is still unexplored when the robot reaches b_i; see Figure 2.7. Since neither of the shortest paths nor the cut between v_i and b_i can be penetrated by the polygon's boundary, the claim follows. ▯

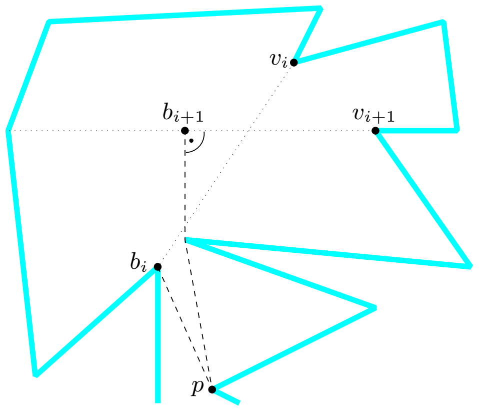

FIG. 2.7. *As seen from* p, *the shortest path to* b_i *runs to the left of the shortest path to* b_{i+1}.

Now we turn to analyzing the length of the path the robot spends on exploring a group of vertices.

LEMMA 2.4. *The robot's path between two consecutive base points is at most* 3 *times as long as the shortest path.*

Proof. Let us call the base points b_1 and b_2, and let c be the point where $cut(v_2)$ is reached. Then c is also the cut's closest point to b_1, by Lemma 2.1. By Lemma 2.2, the robot's path to c is not longer than twice the length of the shortest path from b_1 to c and therefore is also not longer than twice the length of the shortest path from b_1 to b_2; see Figure 2.8.

It remains to account for the walk along the cut from c to b_2. This line segment can be orthogonally projected onto the shortest path from b_1 to b_2 and, therefore, it must be shorter.

Observe that Figure 2.8 is in fact generic: As seen from s, the shortest path to b_1 runs to the left to the shortest path to b_2, by Lemma 2.3; base point b_1 must be located below the cut of v_2; the shortest paths from b_1 to c and from s to b_2 cannot cross because they are both shortest paths to this cut. ▯

The next steps consist of comparing the length of an *ExploreRightGroup* tour with the relative convex hull of the base points visited.

LEMMA 2.5. *The length of a path caused by a call to* ExploreRightGroup *does not exceed* $3\sqrt{2}$ *times the perimeter of the relative convex hull of the base points visited.*

Proof. Let sp be the stage point and $sp = b_0, \ldots, b_{m-1}, b_m = sp$ be the sequence of base points visited by *ExploreRightGroup*. Due to Lemma 2.3, $b_1, \ldots, b_{m-1}$ appear in clockwise order as leaves of the shortest path tree from sp to $b_1, \ldots, b_{m-1}$. So, even

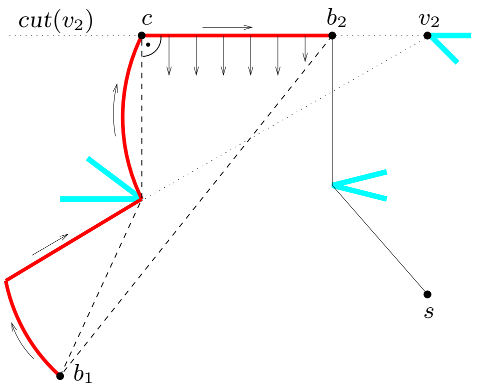

FIG. 2.8. *Line segment cb_2 must be shorter than the shortest path from b_1 to b_2.*

factor 3 of Lemma 2.4 would apply if all base points b_i were vertices of their relative convex hull, $\mathcal{RCH}$, in P. In the following we show how to reduce to this case.

Suppose that for $i \leq k-2$ the base points b_i and b_k are situated on the boundary of $\mathcal{RCH}$ and the points $b_{i+1}, \ldots, b_{k-1}$ in between are not. The shortest path from sp to the cut of each of them must have a right angle to the cut because of Lemma 2.1; see Figure 2.9.

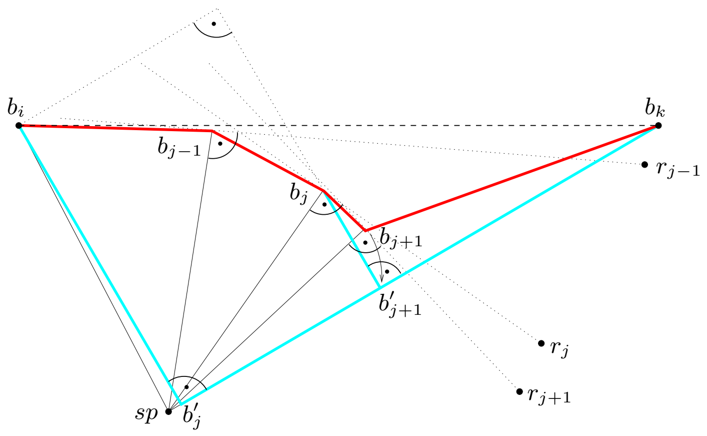

FIG. 2.9. *The cut of vertex r_j containing the base point b_j must not intersect the shortest path from sp to b_{j-1}.*

For $i < j < k$, the cut of b_j must pass above b_{j-1} because otherwise its cut would intersect every possible path from s to b_{j-1}, in particular the robot's path, contradicting the fact that vertex r_j is explored *after* r_{j-1}.

While maintaining these properties, we move the base points one after the other such that the path $b_i, \ldots, b_k$ becomes even longer. For all vertices v of this path, starting with b_{k-1} and going back to b_{i+1}, we do the following. If the path from b_i to b_k makes a left turn at v, like at b_{j+1} in Figure 2.9, then we move v to the point on the shortest path from sp to v's successor such that the left turn is a right angle; see b'_{j+1}. Note that every left turn must be an obtuse angle which again is due to the fact that the cut of b_j must pass above b_{j-1}. In case of a right turn we do nothing. Eventually, we end up with a path whose left turns are all right angles.

Now a maximal sequence of right turns, in the example the chain from b_i to b'_{j+1}, can be replaced by one left turn of 90° which is clearly longer than the chain, see b'_j and the rectangle with vertices b_i, b'_j, and b'_{j+1}. Finally, no right turn remains and the new path makes only one left turn of 90° for which the claim is obvious. ∎

In relating the robot's path to the optimum watchman tour, the following lemma is crucial.

LEMMA 2.6. *All base points are contained in the angle hull $\mathcal{AH}(W_{opt})$.*

Proof. A base point b is, by definition, the closest point to s of a cut. The optimum watchman tour W_{opt} connects the start point s to the cut. Let E be the last edge of the shortest path from s to the cut, i.e., to b.

In most cases, edge E is orthogonal to the cut. Then we have a right angle at b whose one side goes along the cut and touches W_{opt}, while the other side K extends edge E. Either the other endpoint of E equals s, so that K touches W_{opt} in s, or K separates s and the cut because P is simple, and W_{opt} must also be touched by K.

In the remaining case, when there is no right angle between edge E and the cut, point b must be one endpoint of the cut, i.e., it is the target vertex itself or the other endpoint. In both cases the inner angle between E and the cut is necessarily greater than 90°, otherwise there would be a shorter path to the cut. As before either E or its extension meets W_{opt}. ∎

2.3. Subdividing the polygon. Now we want to combine the exploration of several groups of vertices to finally explore the whole polygon P. This is done by making the *ExploreGroup*-procedures recursive.

```
PROCEDURE ExploreRightGroupRec ( in TargetList );
  ExploreRightGroup ( TargetList, ToDoList );    (* ToDoList gets filled in. *)
  Clean up ToDoList:
    retain only those right vertices in ToDoList
    which are highest up in the shortest path tree;
  for all vertices $v$ of ToDoList in clockwise order do
    walk on the shortest path to $v$;    (* connect stage points *)
    ExploreLeftGroupRec( {all known left descendants of $v$ in counterclw. order} );
end ExploreRightGroupRec;
```

The task of *ExploreRightGroupRec* is to explore, from the current position *CP*, all vertices in the input parameter *TargetList* and everything behind.

ExploreRightGroupRec performs in three steps. The *TargetList* is handed over to *ExploreRightGroup*, so *CP* is the new stage point, and after the exploration we are back at this point. We are given a *ToDoList* of candidates for stage points in recursive explorations.

The next step is a necessary cleanup for the *ToDoList*, which contains all purely right descendants of the current stage point which have left children. Some of these right vertices are descendants of others in this list; they must be removed from the list. Only maximal (highest up) right vertices are retained; these will become stage points in further steps. To each of these future stage points we associate a list of all known left descendants that were referenced in *ExploreRightVertex* and *ExploreRightGroup*.

Finally, the remaining vertices in *ToDoList* are visited in clockwise order, at each vertex procedure *ExploreLeftGroupRec* is called to explore the list of all known left descendants (as *TargetList*) from there. *ExploreLeftGroupRec* is the symmetric counterpart of *ExploreRightGroupRec* with one particularity. In the **for** loop, the vertices in *ToDoList* are also visited in clockwise order. The reason for this will become clear in the proof of Theorem 2.10.

To conclude the bottom-up presentation of our strategy, we show the main program. Its task is, of course, to explore a given polygon, P, starting at a boundary point, s. First, in a call to the nonrecursive *ExploreRightGroup*, the right vertices visible from s are explored. The next target list contains all left children of the right vertices just explored and the left vertices visible from s. All these, and everything behind, gets explored by a call to the recursive *ExploreLeftGroupRec* with this target list.

```
PROCEDURE ExplorePolygon ( in $P$, in $s$ );
    ExploreRightGroup ( {clockwise list of all right vertices visible from $s$}, ToDoList );
    TargetList := {all left children of the vertices of ToDoList};
    Add all left vertices visible from $s$ into TargetList and sort counterclockwise;
    ExploreLeftGroupRec ( TargetList );
end ExplorePolygon;
```

Each call of *ExploreRightGroup* or *ExploreLeftGroup* generates a set of base points; the first base point of the set is the stage point. For estimating the length of the complete tour, we distribute all these sets into three categories.

The set of base points generated by the call of *ExploreRightGroup* in *ExplorePolygon* belongs to category 0. For the remaining sets, we use their level of recursion to determine their category: the set of base points generated by a call of *ExploreRightGroup* or *ExploreLeftGroup* at total recursion depth i belongs to category (i mod 3).

For example, the very first call of *ExploreLeftGroup* belongs to category 1, and the calls of *ExploreRightGroup* one level deeper belong to category 2. All calls of *ExploreLeftGroup* have an odd level, and all calls of *ExploreRightGroup* an even level.

A key observation is that two sets of base points of the same category will be mutually invisible; see Lemma 2.7 below. Thus the three categories will contribute a factor of 3 to the final analysis in Theorem 2.10 below.

One might wonder why the first nonrecursive call of *ExploreRightGroup* in procedure *ExplorePolygon* is necessary. Without this call the right vertices visible from s, and everything behind, would not be explored because procedure *ExploreLeftGroupRec* deals exclusively with left vertices visible from s and recursively with their offspring.

LEMMA 2.7. *The relative convex hulls of two sets of base points of the same category are mutually invisible, with a possible exception for their stage points.*

Proof. The recursion depths of two sets of base points, B_1 and B_2, of the same category differ by a multiple of 3, possibly 0, as explained above. Let $s_1 \neq s_2$ be the stage points of B_1, resp., B_2. We distinguish two cases depending on the shortest paths from s to s_1 and s_2.

If stage point s_1 is not on the shortest path from s to s_2 and vice versa then let s_0 be the vertex where the shortest paths from s to s_1 and s_2 separate. Without loss of generality we assume that s_2 is a left vertex and that the clockwise order on the boundary is s_0, s_1, s_2. The left picture in Figure 2.10 shows such a situation.

The base points of B_2 are on cuts of right vertices whose shortest paths from s_0 all pass through s_2. Therefore, the shortest path from s_0 to s_2 is invisible from any point of B_2, except s_2. But this shortest path separates B_2 from B_1, they are therefore mutually invisible, except for s_1 and s_2. This argument easily extends to the convex hulls as well.

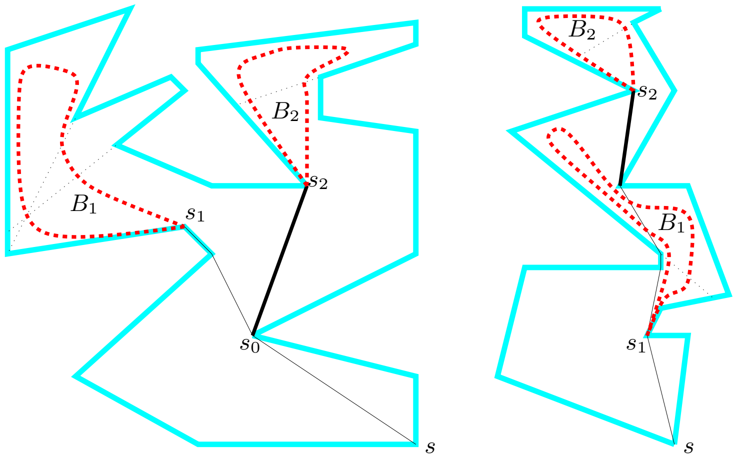

FIG. 2.10. *Base points in B_2 cannot see B_1, two cases.*

Otherwise assume that s_1 lies on the shortest path from s to s_2, see the right picture in Figure 2.10. Then the recursion depths of B_1 and B_2 differ by at least three. Similar to the previous case, no point of B_2 can see the shortest path from s_2 to its parent stage point, but this path definitely separates B_1 and B_2.

Note that a difference of three levels is really necessary. If B_2 is only two levels deeper than B_1 then the stage points s_1 and s_2 are of the same type and the parent stage point of s_2 can be a direct descendant of s_1, and therefore it can very well be contained in $\mathcal{RCH}(B_1)$. □

As a consequence, we conclude that the union of all base points of one category has no shorter perimeter than the perimeters of all of its sets of base points together. Let per($\mathcal{RCH}(A)$) denote the perimeter of the relative convex hull of set A.

LEMMA 2.8. *Let B_1 and B_2 be two sets of base points of the same category. Then we have per($\mathcal{RCH}(B_1)$) + per($\mathcal{RCH}(B_2)$) $\leq$ per($\mathcal{RCH}(B_1 \cup B_2)$).*

As a consequence, we can estimate the path length caused by all calls of *Explore-RightGroup* or *ExploreLeftGroup* in the same category.

LEMMA 2.9. *The path length caused by all calls of ExploreRightGroup and ExploreLeftGroup in one category is less than $6\sqrt{2} \leq 8.5$ times the length of W_{opt}.*

Proof. Let the category consist of sets B_i, $i = 1, \ldots,$ of base points. By Lemma 2.5, the length of the path created by one call of *ExploreRightGroup* or *Explore-LeftGroup* with set B_i is not greater than $3\sqrt{2}\,\mathrm{per}(\mathcal{RCH}(B_i))$.

The relative convex hulls $\mathcal{RCH}(B_i)$ are mutually invisible (Lemma 2.7); hence we conclude from Lemma 2.8 for the path length, L, caused by all calls of *ExploreRight-Group* or *ExploreLeftGroup* of this category,

$$L \leq 3\sqrt{2} \sum_i \mathrm{per}(\mathcal{RCH}(B_i)) \leq 3\sqrt{2}\,\mathrm{per}\left(\mathcal{RCH}\left(\bigcup_i B_i\right)\right).$$

All base points considered are contained in the angle hull of W_{opt}, as Lemma 2.6 has shown, hence the perimeter of their relative convex hull is shorter than the perimeter of $\mathcal{RCH}(\mathrm{AH}(W_{opt}))$.

The perimeter of $\mathcal{RCH}(\mathrm{AH}(W_{opt}))$ is not longer than the perimeter of the angle hull of W_{opt} itself. By Theorem 3.5 this is not greater than twice the length of W_{opt} and the claim follows. □

As the main result for our complete strategy, we obtain a factor of 26.5.

THEOREM 2.10. *For a polygon, P, and a start point s on the boundary of P, a procedure call ExplorePolygon(P, s) explores the polygon and returns to s. The total path length used is less than $(18\sqrt{2}+1) \leq 26.5$ times the length of the optimum watchman tour from s.*

Proof. Since we have three categories of base point sets, all *ExploreRightGroup* and *ExploreLeftGroup* calls together cause a path length of less than $3 \cdot 6\sqrt{2}|W_{opt}|$.

It remains to bound the path length caused by the walks during the **for** loops of *ExploreRightGroupRec* and *ExploreLeftGroupRec*. They connect only stage points by shortest paths, and all those stage points are visited in clockwise order along the boundary of P, independently of whether this is done in *ExploreRightGroupRec* or *ExploreLeftGroupRec*.

The optimum watchman path visits all stage points, and some of them even twice. In any case the sequence of stage points as they appear on the boundary of P is a subsequence of the boundary points of P that are visited by W_{opt} because W_{opt} can not properly cross itself. Therefore, we can be sure that all those walks together make up for an additional path length of at most $|W_{opt}|$. □

3. The angle hull. In on-line navigation algorithms for autonomous robots, analyzing the length of the robot's path is often a complicated issue. Sometimes, only the discovery of certain structural properties has led to a reasonably sharp analysis; see [1, 18, 19, 20] and Rote [24].

Here we provide a new result of this type. It is crucial in analyzing our on-line strategy of section 2, and it seems also to be interesting in its own right.

For convenience, we repeat the definition of the angle hull. Let D be a simply connected region contained in a simple polygon P, then the angle hull, $\mathcal{AH}(D)$, of D with respect to P consists of all points of P that can see two points of D at a right angle. Note that D itself is included in $\mathcal{AH}(D)$. The boundary of the angle hull was denoted the photographer's path of D in section 2.1; for an example see Figure 2.5.

We are now going to analyze the length of the photographer's path, i. e., the perimeter of the angle hull. In section 3.1 we show that, in the indoor setting, the angle hull may have twice the perimeter of D, in the limit. Then, in section 3.2, we prove that this is the worst that can happen.

3.1. The lower bound. We start with the proof that the angle hull of a set D contained in a polygon P can be twice as long as the perimeter of D. Our construction is rather simple, D is a line segment and P is a jagged halfcircle. Region D is called *relatively convex* iff $D = \mathcal{RCH}(D)$.

LEMMA 3.1. *Let $\varepsilon > 0$. There is a polygon, P, and a relatively convex region, D, inside P, for which the boundary of the angle hull $\mathcal{AH}(D)$ with respect to P is longer than $2 - \varepsilon$ times the boundary of D.*

Proof. As our region D, we take a horizontal line segment of length 1. Let p_0, $\dots$, p_n be equidistant points on the halfcircle spanned by D, where p_0 and p_n are the endpoints of D; see Figure 3.1. From each point p_i we draw the right angle to the endpoints of D. Let P be the concatenation of the upper envelope of these angles and its reflection at D. Then we have $P = \mathcal{AH}(D)$ by construction. Let us analyze the upper envelope.

We will show that the length of the jagged line from p_0 to p_n is less than 2, but comes arbitrarily close to 2, as n increases. Let q_i be the intersection of the segments $p_0\,p_{i+1}$ and $p_i\,p_n$. If we rotate, for all i, the ascending segments $q_i\,p_{i+1}$ about p_0 onto D (see the dotted arcs in Figure 3.1), these segments cover disjoint pieces of D,

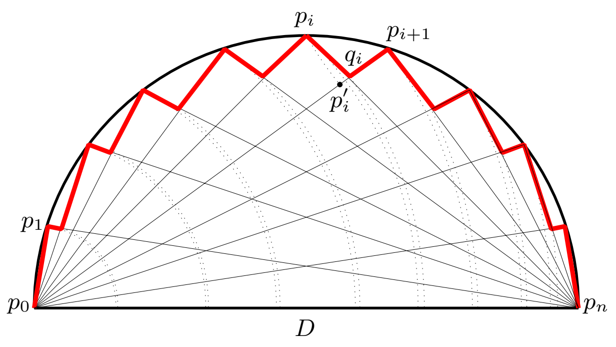

FIG. 3.1. *The boundary of the upper envelope of the right angles is less than* $2|D|$.

so the total length of all ascending segments is always less than 1. By symmetry, the same bound holds for the descending segments. It remains to show that the ascending length can come arbitrarily close to 1.

Consider the triangle $p_i\, q_i\, p_i'$, where p_i' is the orthogonal projection of p_i onto $p_0\, q_i$. Point p_0 is closer to p_i' than to p_i, so for the distances from p_0 to p_i and to q_i we have

$$|p_0\, q_i| - |\, p_0\, p_i| \leq |\, p_0\, q_i| - |\, p_0\, p_i'| = |p_i'\, q_i| = |p_i\, q_i| \sin\frac{\pi}{2n}.$$

The total length of all ascending segments is therefore 1 minus the following rest:

$$\sum_i (|p_0\, q_i| - |\, p_0\, p_i|) \leq \sin\frac{\pi}{2n} \sum_i |p_i\, q_i| \leq \sin\frac{\pi}{2n}.$$

For $n \to \infty$, this tends to 0. The last inequality holds because $\sum_i |p_i\, q_i| \leq 1$ is the length of all descending segments. $\square$

The proof also works for nonequidistant points as long as the maximum distance between subsequent points tends to 0. We are obliged to Seidel [26] for this elegant proof of Lemma 3.1.

3.2. The upper bound. Interestingly, the same jagged lines as used in the proof of Lemma 3.1 are also very useful in the proof of the upper bound. For any circular arc C we can construct a jagged line by distributing auxiliary points along C and by taking the upper envelope of the right angles at these points whose sides pass through the two spanning vertices of C; see Figure 3.2. We denote with *jagged length*, $J(C)$, of C the limit of the lengths of these jagged lines as the maximum distance between subsequent points tends to 0. This limit is well defined, i.e., it does not depend on how the points are chosen. In the proof of Lemma 3.1 we have already seen how to determine this length by separately estimating the lengths of the ascending and descending segments. For the jagged length of a circular arc with diameter 1 from angle α to angle β (see Figure 3.2), we obtain analogously to the proof of Lemma 3.1

$$J(C) = \sin\beta - \sin\alpha - \cos\beta + \cos\alpha,$$

which can also be written as

$$J(C) = \int_\alpha^\beta (\cos\gamma + \sin\gamma)\, d\gamma\,.$$

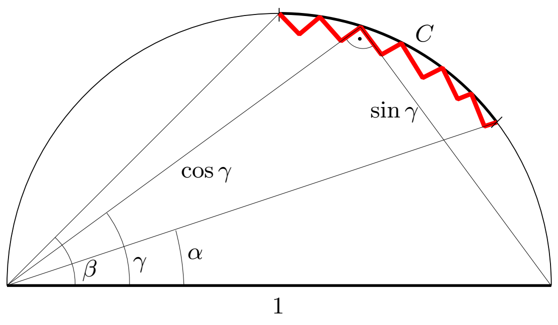

FIG. 3.2. *Analyzing the jagged length, $J(C)$, of a circular arc C.*

LEMMA 3.2. *The jagged length of an arc is always greater than the arc length itself.*

Proof. Consider a circle with diameter d and a circular arc a on its boundary. Two lines from an arbitrary point on the boundary through the endpoints of the arc always intersect in the same angle ϕ, by the generalized Thales's theorem. For the length of a, we have $|a| = \phi d$; see Figure 3.3.

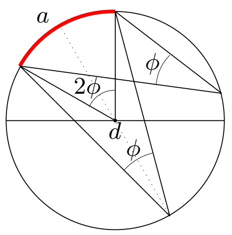

FIG. 3.3. $|a| = \phi d$.

So the arc length of the arc C in Figure 3.2 equals $\beta - \alpha$, and we have

$$J(C) = \int_{\alpha}^{\beta} (\cos\gamma + \sin\gamma)\, d\gamma \geq \int_{\alpha}^{\beta} 1 \, d\gamma = \beta - \alpha\,;$$

the inequality follows from $\cos\gamma + \sin\gamma \geq 1$ for $\gamma \in [0, \frac{\pi}{2}]$. ☐

The integral form for the jagged length also has a geometric interpretation. Let us consider a right angle with slope γ contained in the halfcircle, as shown in Figure 3.2. The length of the two sides of the right angle equals $\cos\gamma + \sin\gamma$. If we define

$$C_\gamma := \begin{cases} \text{length of the right angle} & \text{if its apex is contained in } C, \\ 0 & \text{otherwise,} \end{cases}$$

we obtain the nice form

$$J(C) = \int_{0}^{\frac{\pi}{2}} C_\gamma \, d\gamma\,.$$

This form is used in the proof of the next lemma.

LEMMA 3.3. *Let D be a line segment, and let P be a surrounding polygon such that P and the angle hull $\mathcal{AH}(D)$ with respect to P touch only at vertices of P; see*

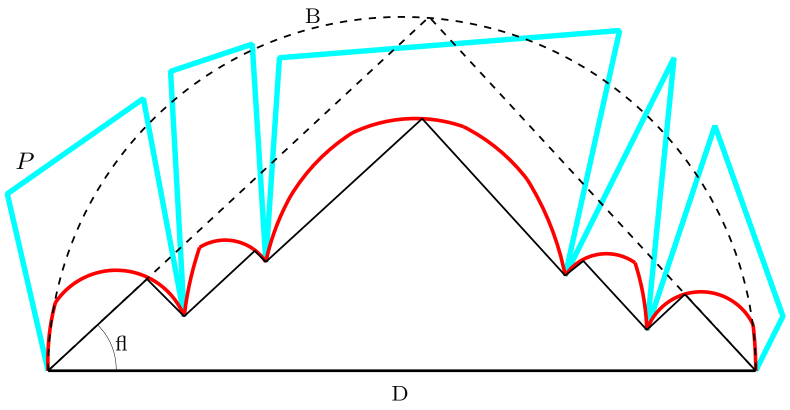

FIG. 3.4. *For a line segment D we have $J(\mathcal{AH}(D)) = J(B) = 2|D|$.*

Figure 3.4. *Then the arc length of $\mathcal{AH}(D)$ with respect to P from one endpoint of D to the other is less than* $2|D|$.

Proof. By Lemma 3.2, the arc length of $\mathcal{AH}(D)$ is certainly shorter than the jagged length of $\mathcal{AH}(D)$, i.e., the sum of the jagged lengths of all circular arcs of $\mathcal{AH}(D)$, and we obtain

$$\mathit{length}(\mathcal{AH}(D)) \le J(\mathcal{AH}(D)) = \sum_{C\in\mathcal{AH}(D)} J(C)$$

$$= \sum_{C\in\mathcal{AH}(D)} \int_0^{\frac{\pi}{2}} C_\gamma \, d\gamma = \int_0^{\frac{\pi}{2}} \left(\sum_{C\in\mathcal{AH}(D)} C_\gamma \right) d\gamma \,.$$

But for any angle γ the sum over the lengths of the right angles of slope γ which are contained in the halfcircles of the different circular arcs of $\mathcal{AH}(D)$ is equal to the length, B_γ, of the big right angle in the halfcircle B spanned by the two endpoints of D, which means that

$$\int_0^{\frac{\pi}{2}} \left(\sum_{C\in\mathcal{AH}(D)} C_\gamma \right) d\gamma = \int_0^{\frac{\pi}{2}} B_\gamma \, d\gamma = J(B) = 2|D| \,. \qquad \square$$

Note that in the proof of Lemma 3.3 the halfcircle B does not depend on P, and $J(\mathcal{AH}(D)) = J(B)$ therefore means that the jagged lengths of the angle hulls of D for different surrounding polygons P are all identical! We may also say that we have bounded the length of the angle hull with respect to a surrounding polygon P by the jagged length of the angle hull without obstacles.

LEMMA 3.4. *The statement of Lemma* 3.3 *remains true if D is a convex chain instead of a line segment.*

Proof. We consider a convex chain, D, and a surrounding polygon, P, such that P and the angle hull $\mathcal{AH}(D)$ with respect to P touch only at vertices of P.

We make a construction similar to the proof of Lemma 3.3. For an angle γ we find the tangent to D with that slope. Starting with the touching vertex we go into direction γ until we hit an arc of the angle hull, then we turn by a right angle and go to the vertex of P (or D) that co-spans the current arc. Here we turn back to the original direction and continue accordingly to obtain a connected chain of right angles; see Figure 3.5.

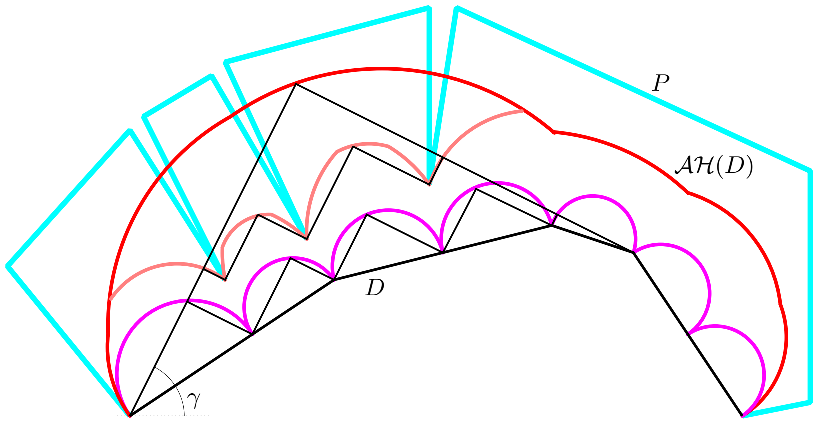

FIG. 3.5. *Three chains of right angles which are all of the same length.*

This chain has the same length as the two sides of the "unfolded" big right angle of slope γ which generates the angle hull without obstacles. As before, this shows that the jagged length of the angle hull does not depend on P.

Now it is not difficult to see how long it really is. Consider the set of halfcircles spanned by the segments of D. Analogously to the previous construction, we can construct the chain of right angles of slope γ below these halfcircles which again has the same length. But these right angles represent the jagged lengths of the isolated segments, and each of them equals twice the length of the segment, by Lemma 3.3. Therefore the total length of the angle hull of D is less than twice the length of D. ∎

To obtain our main result we need to consider an arbitrary surrounding polygon P that influences the angle hull not only with acute reflex vertices but also with its edges.

THEOREM 3.5. *Let P be a simple polygon containing a relatively convex polygon D. The arc length of the boundary of the angle hull, $\mathcal{AH}(D)$, with respect to P is less than 2 times the length of D's boundary. This bound is tight.*

The bound also holds if there are several obstacles instead of P that influence the angle hull and also if their boundaries consist of arbitrary curves.

Proof. Each convex chain of D can be treated separately because the angle hull must pass through the reflex vertices of D.

First, we consider the angle hull $\mathcal{AH}_1(D)$ with respect to only the vertices of P as obstacle points. Its arc length is less than $2|D|$, by Lemma 3.4.

Now also the edges come into play. The angle hull $\mathcal{AH}_2(D)$ with respect to the whole of P contains circular arcs and some pieces of P's edges, for an example see Figure 2.5. The circular arcs of $\mathcal{AH}_2(D)$ are also part of $\mathcal{AH}_1(D)$.

For every piece of an edge which contributes to $\mathcal{AH}_2(D)$, the piece's two endpoints are also on the boundary $\mathcal{AH}_1(D)$. Therefore, $\mathcal{AH}_2(D)$ can only be shorter than $\mathcal{AH}_1(D)$.

The bound is tight by Lemma 3.1.

The proof easily generalizes to the case of several obstacles around D that influence the angle hull instead of P. Indeed, we have never used the fact that the parts of P that are touching the angle hull are connected by edges of P. And if we have several obstacles, we can always connect them to a single one by edges which do not influence the angle hull.

The proof carries over to arbitrary curves by approximation of these curves with polygons. ▯

The following variation of Theorem 3.5 for an "incomplete angle hull" is used for analyzing the exploration strategy in section 2.

COROLLARY 3.6. *Consider a convex chain, D, from s to t and its angle hull from s to some point g. The jagged length of this part of the angle hull is bounded by twice the length of the shortest path around D from s to g.*

Proof. Let m be the last segment of D and let α be the angle between m and the last segment of the shortest path from s to g; see Figure 3.6.

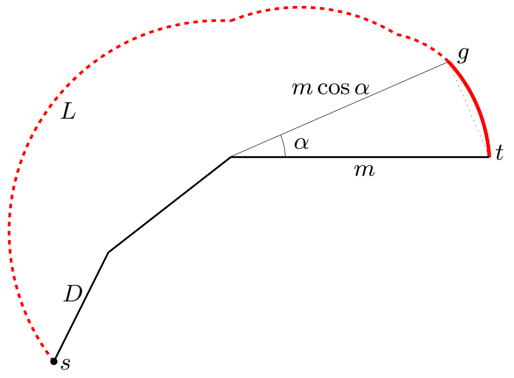

FIG. 3.6. *The cut of r_1 is reached at point g.*

The jagged length of the angle hull from g to t equals $m(1 + \sin\alpha - \cos\alpha)$. The path, L, from the base point to g can therefore be estimated in the following way, using Theorem 3.5:

$$\begin{aligned} L &\le 2|D| - m(1 + \sin\alpha - \cos\alpha) \\ &= 2(|D| - m + m\cos\alpha) + m(1 - \cos\alpha - \sin\alpha) \\ &\le 2(|D| - m + m\cos\alpha). \end{aligned}$$

The last inequality holds because of $0 \le \alpha \le \frac{\pi}{2}$. But $|D| - m + m\cos\alpha$ is exactly the length of the shortest path from the base point to g. ▯

Remark. The result of Theorem 3.5 can be generalized to angles $\phi \ne 90°$. Namely, for the jagged length of a circular arc C spanned with fixed angle ϕ by a chord of length 1 from angle α to angle β we have

$$J_\phi(C) = (\sin\beta - \sin\alpha - \cos\beta + \cos\alpha)\frac{\cos\phi + 1}{\sin^2\phi}$$

from which a tight factor of $2(\cos\phi + 1)/\sin^2\phi$ follows; see [16] for this and other interesting extensions. Interestingly, the angle hull with respect to an arbitrary angle has independently been described, in a different context and without estimations about its length, by de Berg et al. [10]—this corresponds to our "outdoor setting"—and in the successor article by Cheong and van Oostrum [8] for the "indoor setting," i.e., with obstacles.

4. Conclusions. We have seen that a combination of suitable analysis techniques is necessary for proving an upper bound for the competitive factor of a rather simple strategy. Still, we believe that its actual performance, even in the worst case, is considerably better than the proven bound. Establishing a lower bound for polygon exploration, higher than the trivial $(1+\sqrt{2})/2 \approx 1.207$, and closing the gap to the upper bound seem to be challenging problems.

There are many interesting variations and generalizations of the polygon exploration problem. For example, one could study different cost models for the robot's motion. Also, the case of polygons with holes deserves investigation. Here the off-line problem becomes NP-hard, by reduction from the traveling salesperson problem. Recently, Albers, Kursawe, and Schuierer [2] have shown that in a rectilinear environment no better competitive factor than $O(\sqrt{k})$ can be achieved for the on-line problem in the presence of k rectilinear holes, what was known before only for general polygons.

We have also introduced a new type of hull operator that suits us well in analyzing the on-line exploration strategy and that is interesting in its own right. Here we have analyzed the perimeter of the angle hull, $\mathcal{AH}(D)$, in terms of the perimeter of the region D. A number of interesting questions remain open: If we consider a subset of D, is the perimeter of its angle hull always shorter than the perimeter of $\mathcal{AH}(D)$? Does the iterated construction of the angle hull approximate a circle? How can angle hulls be generalized, and analyzed, in three dimensions?

Acknowledgment. We would like to thank the anonymous referees for their valuable comments.

REFERENCES

[1] O. Aichholzer, F. Aurenhammer, C. Icking, R. Klein, E. Langetepe, and G. Rote, *Generalized self-approaching curves*, in Proceedings of the 9th Annual International Symposium on Algorithms and Computation, Lecture Notes in Comput. Sci. 1533, Springer-Verlag, Berlin, 1998, pp. 317–326.

[2] S. Albers, K. Kursawe, and S. Schuierer, *Exploring unknown environments with obstacles*, in Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, 1999, pp. 842–843.

[3] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, *Approximation Algorithms for Lawn Mowing and Milling*, Tech. Report, Mathematisches Institut, Universität zu Köln, 1997.

[4] P. Berman, *On-line searching and navigation*, in On-line Algorithms: The State of the Art, A. Fiat and G. Woeginger, eds., Springer-Verlag, Berlin, 1998, pp. 232–241.

[5] A. Blum, P. Raghavan, and B. Schieber, *Navigating in unfamiliar geometric terrain*, SIAM J. Comput., 26 (1997), pp. 110–137.

[6] S. Carlsson and H. Jonsson, *Computing a shortest watchman path in a simple polygon in polynomial time*, in Proceedings of the 4th Workshop on Algorithms and Data Structures, Lecture Notes in Comput. Sci. 955, Springer-Verlag, Berlin, 1995, pp. 122–134.

[7] S. Carlsson, H. Jonsson, and B. J. Nilsson, *Finding the shortest watchman route in a simple polygon*, in Proceedings of the 4th Annual International Symposium on Algorithms and Computation, Lecture Notes in Comput. Sci. 762, Springer-Verlag, Berlin, 1993, pp. 58–67.

[8] O. Cheong and R. van Oostrum, *Reaching a polygon with directional uncertainty*, Internat. J. Comput. Geom. Appl., (2001), to appear.

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

[10] M. de Berg, L. Guibas, D. Halperin, M. Overmars, O. Schwarzkopf, M. Sharir, and M. Teillaud, *Reaching a goal with directional uncertainty*, Theoret. Comput. Sci., 140 (1995), pp. 301–317.

[11] X. Deng, T. Kameda, and C. Papadimitriou, *How to learn an unknown environment*, in Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science, Los Alamitos, CA, 1991, pp. 298–303.

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

[13] A. Fiat and G. Woeginger, eds., *On-line Algorithms: The State of the Art*, Lecture Notes in Comput. Sci. 1422, Springer-Verlag, Berlin, 1998.

[14] M. Hammar and B. J. Nilsson, *Concerning the time bounds of existing shortest watchman route algorithms*, in Proceedings of the 11th International Symposium on Fundamentals of Computation Theory, Lecture Notes in Comput. Sci. 1279, Springer-Verlag, Berlin, 1997, pp. 210–221.

[15] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel, *A competitive strategy for learning a polygon*, in Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, 1997, pp. 166–174.

[16] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel, *Moving an angle around a region*, in Proceedings of the 6th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Comput. Sci. 1432, Springer-Verlag, Berlin, 1998, pp. 71–82.

[17] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel, *The polygon exploration problem: A new strategy and a new analysis technique*, in Robotics: The Algorithmic Perspective, Proceedings of the 3rd Workshop on Algorithmic Foundations of Robotics, A.K. Peters, Wellesley, MA, 1998, pp. 211–222.

[18] C. Icking and R. Klein, *Searching for the kernel of a polygon: A competitive strategy*, in Proceedings of the 11th Annual ACM Symposium on Computational Geometry, ACM, New York, 1995, pp. 258–266.

[19] C. Icking, R. Klein, and E. Langetepe, *An optimal competitive strategy for walking in streets*, in Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Comput. Sci. 1563, Springer-Verlag, Berlin, 1999, pp. 110–120.

[20] C. Icking, R. Klein, and L. Ma, *How to look around a corner*, in Proceedings of the 5th Canadian Conference on Computational Geometry, University of Waterloo, Waterloo, ON, Canada, 1993, pp. 443–448.

[21] J. S. B. Mitchell, *Geometric shortest paths and network optimization*, in Handbook of Computational Geometry, J.-R. Sack and J. Urrutia, eds., Elsevier, North-Holland, Amsterdam, 2000, pp. 633–701.

[22] S. Ntafos, *Watchman routes under limited visibility*, in Proceedings of the 2nd Canadian Conference on Computational Geometry, University of Ottawa, Ottawa, ON, Canada, 1990, pp. 89–92.

[23] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar, *Robot Navigation in Unknown Terrains: Introductory Survey of Non-heuristic Algorithms*, Tech. Report ORNL/TM-12410, Oak Ridge National Laboratory, Oak Ridge, TN, 1993.

[24] G. Rote, *Curves with increasing chords*, Math. Proc. Cambridge Philos. Soc., 115 (1994), pp. 1–12.

[25] J.-R. Sack and J. Urrutia, eds., *Handbook of Computational Geometry*, North-Holland, Amsterdam, 2000.

[26] R. Seidel, *private communication,* 1997.

[27] D. D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules*, Commun. ACM, 28 (1985), pp. 202–208.

[28] X. Tan and T. Hirata, *Constructing shortest watchman routes by divide-and-conquer*, in Proceedings of the 4th Annual International Symposium on Algorithms and Computation, Lecture Notes in Comput. Sci. 762, Springer-Verlag, Berlin, 1993, pp. 68–77.

[29] X. Tan, T. Hirata, and Y. Inagaki, *Corrigendum to "An incremental algorithm for constructing shortest watchman routes,"* Internat. J. Comput. Geom. Appl., 9 (1999), pp. 319–323.