Walking an Unknown Street with Bounded Detour

Rolf Klein*

Abstract

A polygon with two distinguished vertices, s and g, is called a street *iff the two boundary chains from s to g are mutually weakly visible. For a mobile robot with on-board vision system we describe a strategy for finding a short path from s to g in a street not known in advance, and prove that the length of the path created does not exceed* $1+\frac{3}{2}\pi$ *times the length of the shortest path from s to g. Experiments suggest that our strategy is much better than this, as no ratio bigger than* 1.8 *has yet been observed. This is complemented by a lower bound of* 1.41 *for the relative detour each strategy can be forced to generate.*

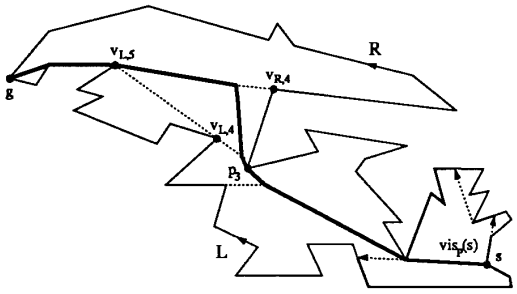

Figure 1: A street P, the visibility polygon of P at s, and the path found by our strategy.

1 Introduction

How to quickly determine the *shortest path* between two points in a simple polygon P is a classical problem in computational geometry. An optimal solution was provided by Guibas and Hershberger [6] by proving that one can, in $O(n)$ preprocessing time, build up a search structure of size $O(n)$ that allows the shortest path between any two points in P to be computed within time $O(\log n + k)$, where n is the number of edges of P and k denotes the number of line segments the shortest path consists of. The preprocessing step requires a triangulation of P; due to Chazelle [2] or Seidel [13], this can be computed in $O(n)$ worst case or randomized time, too.

Such algorithms are based on the assumption that the whole polygon is known in advance. In real life, however, one often has to move through an environment without completely knowing it, but rather on the basis of *local* information provided by acoustical, visual, or tactile sensors. Given the importance of this problem it is quite surprising how few results exist; see e.g. [10, 9, 8, 3, 5], or [11, 12] for further references. Lumelsky and Stepanov [9] studied the case of a mobile robot equipped with a tactile sensor in an environment of obstacles. The robot is given the coordinates of the goal and of its own position in the plane; it starts heading straight to the goal until it hits an obstacle. Then it searches its contour for a point with minimum distance to the goal, and resumes from there. This simple strategy finds a path to the goal, if there is one, and the length of the path is bounded by 1.5 times the sum of the perimeters of all obstacles that are not farther away from the goal than the start point. Papadimitriou and Yanakakis [12] considered scenes of disjoint isothetic rectangles. They were able to bound the lenght of the generated path in terms of the length of the shortest path. Similar bounds were achieved by Eades, Lin, and Wormald [5] for barriers perpendicular to the line connecting start and goal, and by Blum, Raghavan, and Schieber [1] for more general convex obstacles; the latter paper also includes a randomized algorithm for non-convex obstacles. Other recent work is by Deng, Kameda, and Papadimitriou [4].

In this paper we consider the following problem. Let P be a simple planar polygon with a start vertex, s, and a goal vertex, g. Assume that at vertex s a mobile robot is located that wants to get to g on as short a path as possible. The robot is equipped with a vision system that provides, for each point p in P, the *visibility polygon* $vis_P(p)$ of P at p; see Figure 1. The goal, g, is marked so that the robot can recognize it as soon as it sees it.

We do not discuss here the issues of image processing or the computational complexity involved. Rather, we are interested in a general strategy S such that the

*Praktische Informatik VI, FernUniversität–GH–Hagen, Feithstraße 152, D–5800 Hagen, Germany. This work was partially supported by the Deutsche Forschungsgemeinschaft, grant Kl-655, 2-1.

CH3062-7/91/0000/0304$01.00 © 1991 IEEE

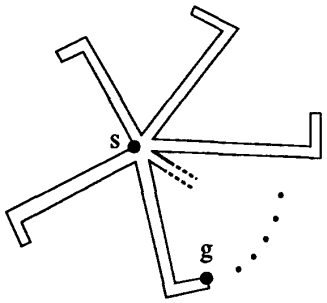

Figure 2: No strategy can guarantee a bounded relative detour in this case.

relative detour

$$D_S(P) := \frac{\text{length of the path created by strategy } S}{\text{length of the shortest path}}$$

becomes as small as possible. Note that just to find the goal represents no problem because the robot could simply follow the boundary until g is encountered (this is what the strategy of [9] mentioned above would in general do.) The difficulty is in keeping the detour small.

With general polygons one cannot hope for the relative detour to be bounded. In Figure 2, for example, there is no way of finding the goal other than by trying the streets leading away from the central "crossing" one by one. Introducing the Euclidean distance between start and goal as an additional parameter, as proposed in [12], does not help. Also, the upper bound to the detour should not depend on the number of vertices of the scene, a parameter introduced in [1], because we want to model smooth scenes, too.

In this paper we study a special class of polygons, based on the following observation. Racetracks and rivers, like the Rhine, contain many curves and bays, but (almost) no cul-de-sacs leading away from the main route. We formalize this property as follows.

Definition 1.1 Let P be a simple polygon with two distinguished vertices, s and g, and let L and R denote the oriented boundary chains leading from s to g. Then P is called a *street* iff L and R are mutually weakly visible, i.e. if each point of L can be seen from at least one point of R, and vice versa.

An example is shown in Figure 1. In a street, the situation depicted in Figure 2 cannot occur.

It follows from Definition 1.1 that each point of L can be connected to some point of R by a line segment contained in the polygon, and vice versa. On its way from s to g the robot is to cross all these line segments, so it sees the whole of L and R. Hence, a short path to the goal also represents a good solution to the *terrain acquisition problem* addressed e.g. in [8].

Our solution consists of two independent parts. In Section 2 we describe a *high-level strategy* that finds a path from s to g subject to the following invariants. At each position p on this path, either the robot can see the goal (then it walks straight towards it), or the robot knows which of the corners visible ahead is visited by the shortest path from s to g (then it walks straight towards this vertex), or the robot can identify two corners ahead one of which must be visited by the shortest path from s to g, but it cannot tell which one; see Figure 5 where v, too, could be the goal. In this case, the robot chooses a point t on the line segment connecting the two vertices and walks straight in direction of this point. How to choose this point is left to a *low-level strategy*. However, it is crucial for the overall length of the generated path *how* this choice is made. There are some suggestive approaches that can result in an unbounded detour; see Section 3.

In Section 3 the low-level strategy *lad* is proposed that tries to minimize the *local absolute detour*, whenever an ambiguity arises in form of two candidate corners. Whereas promising low-level strategies are easily invented, it appears to be quite difficult to analyze them. We prove in Section 4 that for each street P the estimate

$$D_{lad}(P) < 1 + \frac{3}{2}\pi$$

holds. However, experiments show that our approach works much better than this bound suggests; we have not been able to construct a street P with a relative detour $D_{lad}(P) \geq 1.8$. On the other hand, we show in Section 3 that each strategy can be forced to produce a detour of at least $\sqrt{2} = 1.141...$, by choosing a suitably bad street.

Acknowledgement. The author wants to thank Joseph S. B. Mitchell and Günter Rote for helpful discussions during the seminar on computational geometry at Schloß Dagstuhl in October 1990, and Christian Icking for helpful discussions and for conducting the experiments at the UGH Essen.

2 A High Level Strategy

First we state the visibility properties of streets that will be used by the mobile robot on its way. Let P denote a street with start and goal vertices s and g. The polygonal chains L and R are ordered in direction

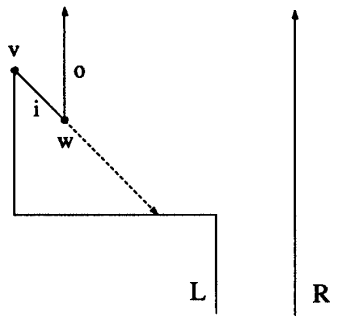

Figure 3: This situation cannot occur in a street.

from s to g. For simplicity, we assume that no three vertices of P are colinear.

Lemma 2.1 *The situation shown in Figure 3 cannot occur in a street. Neither can the prolongation of edge o beyond w hit a point of L ahead of w. The same holds for chain R.*

In fact, in Figure 3, vertex v cannot see any point of R, contradicting Definition 1.1. We have shown in [7] that the conditions stated in Lemma 2.1 are also sufficient for a polygon to be a street. They can be tested in time $O(n \log n)$. Also, from each interior point of P points of L and of R are visible.

The *visibility polygon* $vis_P(p)$ from a point p in P contains the circular list of all pieces of the boundary of P that can be seen from p, called the *umbrella* of p; see Figure 1. Where two pieces meet, the one hit first by the ray from p is said to be *below* the other. Its endpoint is a reflex vertex of P, i.e. one whose internal angle is greater than π.

If the goal is not visible from p, the shortest path to g consists of a line segment leading to such a reflex vertex, followed by a polygonal chain that does not enter $vis_P(p)$ again.

Lemma 2.2 *As one scans the umbrella of $vis_P(p)$, all pieces belonging to L must appear consecutively and in clockwise order around p, whereas pieces of R appear consecutively in counterclockwise order, with respect to the orders on the chains.*

Lemma 2.3 *Suppose that from position p in P an initial piece of the outgoing edge of a reflex vertex v is visible. Then on each path from s to g in P there exists a position from where the incoming edge is visible.*

This fact is crucial. It does, however, not imply that the robot needs to store all parts of P it has seen so far; see Corollary 2.10.

Next, the high-level strategy is listed.

```
PROCEDURE HighLevelStrategy;
CONST     s:   PointOfP;      (* start *)
          g:   PointOfP;      (* goal *)
VAR       p:   PointInP;      (* current position *)
          p':  PointInP;      (* here event occurs *)
      vL, vR:  PointOfP;      (* foremost points on
                                 L and R robot
                                 has so far
                                 identified *)
BEGIN (* HighLevelStrategy *)
  p := s;
  determine vL and vR in visP(p);
  WHILE vL ≠ vR DO            (* g not visible *)
    IF p, vL, vR are colinear
      THEN                    (* Case 1 *)
        p := the closer one of (vL, vR);
        walk straight to p;
        determine vL and vR in visP(p);
      ELSE                    (* Case 2 *)
        choose t in vLvR;
                              (* low-level strategy *)
        walk straight towards t UNTIL
          event occurs at some point p';
        p := p';
        update vL and vR in visP(p)
    END (* IF *)
  END (* WHILE *);
  walk straight to g
END HighLevelStrategy.
```

The path hereby created consists of a chain of line segments $l_1 \ldots l_m$ in P, where $l_i = \overline{p_{i-1}p_i}$, $p_0 = s$, and $p_m = g$. At the start point, p_{i-1}, of each new line segment the robot determines two points, $v_{L,i}$ and $v_{R,i}$, in the part of $vis_P(p_{i-1})$ ahead. These are the robot's orientation marks, enjoying the following properties.

Invariant 2.4 *Assume that the robot has arrived at a point $z \in l_i - \{p_i\}$ where $1 \leq i < m$. Then the following holds.*

1. *Exactly one of the cases shown in Figure 6 applies to l_i.*
2. *So far the robot has seen no part of P ahead of $v_{L,i}$ or $v_{R,i}$ except point c_i in Case 1.2 and segment $c_{i,z}$ in Case 2.*
3. *For all possible prolongations into a street of the part of P the robot has so far seen, $v_{L,i} \in L$ and $v_{R,i} \in R$ hold. The shortest path from s to the goal visits $v_{L,i}$ or $v_{R,i}$. In Case 1, the endpoint of l_i, p_i, lies on the shortest path. In Case 2, $\alpha_i < \pi$ holds.*

As Figure 6 shows, Case 1 of the above algorithm has two subcases. Though 1.2 is but a degenerate

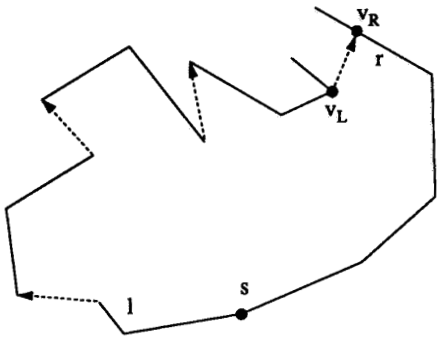

Figure 4: Here the shortest path must visit v_L.

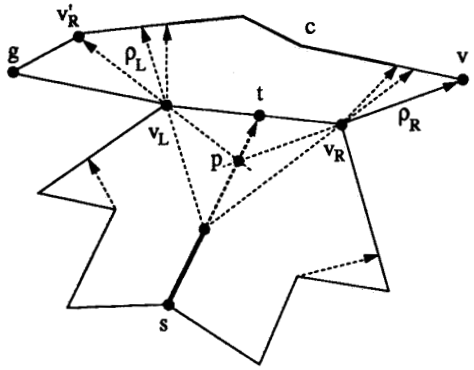

Figure 5: The shortest path visits v_L, but if the goal were at v then it would run through v_R.

case of 2, we subsume it under Case 1 because the next vertex visited by the shortest path to s is known.

It remains to explain *how* the robot determines v_L and v_R. This process is intrinsically incremental, in that the robot would not be able to determine its orientation marks correctly if it were to start from some position in the middle of P (e.g. the i-dot in $vis_P(s)$ in Figure 1).

The construction is based on the following additional invariant. We put $v_{L,0} := v_{R,0} := s$.

Invariant 2.5 *Assume that $i = 0$ holds or that p_i is endpoint of a Case 1 type segment, and that p_i cannot see g. Then the following holds for the pieces in the umbrella of p_i that lie in clockwise order between the old orientation marks $v_{L,i}$ and $v_{R,i}$.*

- *No piece of L is below its left neighbor.*
- *No piece of R is below its right neighbor.*

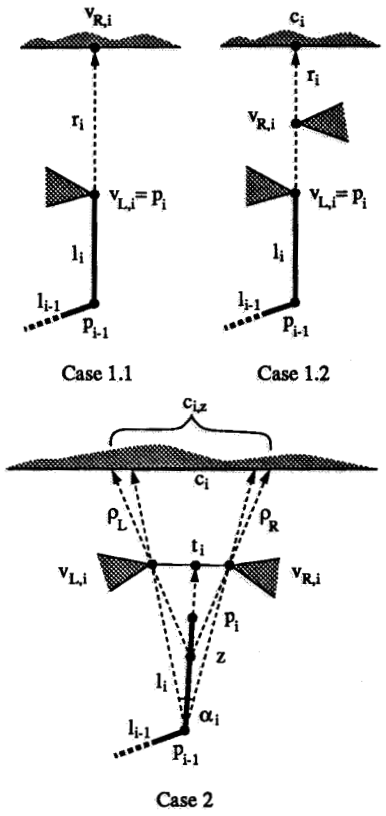

Figure 6: The cases of Invariant 2.4.

Lemma 2.6 *In the situation described in Invariant 2.5 three orderings are possible among the pieces between $v_{L,i}$ and $v_{R,i}$ (including those containing $v_{L,i}$ and $v_{R,i}$.)*

1. *Each piece lies above its left neighbor. Only the rightmost piece, r, belongs to R.*
2. *Each piece lies above its right neighbor. Only the leftmost piece, l, belongs to L.*
3. *There is a unique piece, c, lying above both its neighbors. Its left neighbor belongs to L, its right neighbor to R.*

Proof: Assume there is a piece c that lies above both its neighbors. Its left neighbor is below c, so it must belong to L, due to Lemma 2.3. Similarly, the right neighbor of c belongs to R. Lemma 2.2 implies that there can be at most one piece like c.

If there is no such piece then 1) or 2) must apply because no piece can be below both its neighbors, due to Invariant 2.5. □

For $i = 0$, the first and the third alternative are depicted in Figure 4 and Figure 5, respectively. Here the leftmost and the rightmost piece are joined at $v_{L,0} = v_{R,0} = s$.

Next, we consider a Case 2 type segment l_i, see Figure 6. As the robot moves towards t_i, the rays ρ_L and ρ_R emanating from its current position, z, rotate about their pivots, $v_{L,i}$ and $v_{R,i}$, and segment $c_{i,z}$ grows longer. The following lemma shows that the robot will obtain more information before it arrives at t_i. This event marks the endpoint, p_i, of segment l_i. In the following we drop the index i, and denote $v_{L,i+1}$ by v'_L, etc. The events no. 2-4 are illustrated by the Figures 5, 8 ii), and 9, correspondingly.

Lemma 2.7 *Assume Case 2 applies to v_L and v_R; cf. Figure 6. Then one of the following events occurs* before *the robot arrives at $t \in \overline{v_L v_R}$ or hits the boundary of P.*

1. *The goal becomes visible.*
2. *The growing segment c reaches v_L or v_R.*
3. *An endpoint of c is encountered by one of the rays ρ_L, ρ_R.*
4. *One of the rays is blocked by a reflex vertex.*

Proof: W.l.o.g. we assume that c belongs to chain R, and that s is the robot's current position.

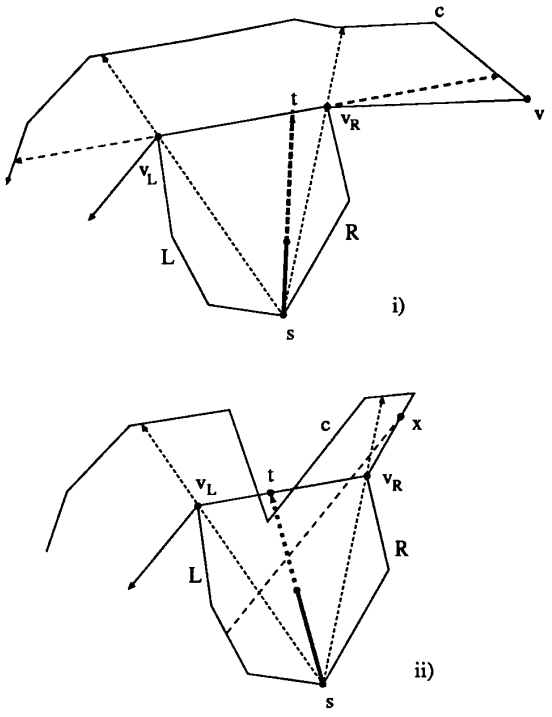

Figure 7: In i), vertex v cannot see any point of L. In ii), the robot crosses the line segment connecting x with L before hitting the boundary.

If t *is visible* from s, and if none of the four events occured then both rays would rotate about theit pivots without being obstructed, until the robot arrives at t on $\overline{v_L v_R}$, from where it can see the grown segment c at the angle π (Figure 8 i) shows that parts of c may become invisible, as the robot proceeds. But the pieces between the hit points of ρ_L and ρ_R are known to belong to the same chain.) This situation is depicted in Figure 7 i); it contradicts Lemma 2.1 because the prolongation of the outgoing edge of v_R hits R ahead of v_R.

If t *is not visible* then t lies outside the polygon because only c can obstruct the view from s to t, but c itself is fully visible from s. In this case the robot must cross each line segment in P connecting a point $x \in R$ between v_R and c with L, before it arrives at c; see Figure 7 ii). Thus, event no. 2 applies. □

Now we describe how the robot determines the new orientation marks v'_L and v'_R on arriving at p.

Determination of v'_L and v'_R

1. p sees g. Let $v'_L := v'_R := g$.
2. $p = s$ or p is endpoint of a Case 1 type segment.
   (a) Lemma 2.6 1) holds. Let v'_R be the left endpoint of piece r, and let v'_R be the vertex of r's left neighbor below v'_R; see Figure 4.
   (b) Lemma 2.6 2) holds. Symmetrically.
   (c) Lemma 2.6 3) holds. Let v'_L and v'_R be the reflex vertices of the neighbors of c; see Figure 5.
3. p is endpoint of a Case 2 type segment. We distinguish between the different events that may have occurred at p.
   (a) Event no. 1. See 1) above.
   (b) Event no. 2 shown in Figure 5. Let v'_R be the hit point of $\rho_L = \overrightarrow{pv_L}$, and $v'_L := v_L$.
   (c) Event no. 3 shown in Figure 8 ii). Let v'_L be the reflex vertex where $\rho_R = \overrightarrow{pv_R}$ slips off c, and $v'_R := v_R$.
   (d) Event no. 4 shown in Figure 9. Let v'_R be the reflex vertex blocking ρ_R, and $v'_L := v_L$.

The symmetric versions of 3 (a), 3(b), and 3(c) are treated similarly.

The algorithm HighLevelStrategy is now completely specified.

Theorem 2.8 *For each $i \geq 0$, Invariant 2.5 holds for p_i, and Invariant 2.4 holds for $l_{i+1} = \overline{p_i p_{i+1}}$. The sequences $(v_{L,i})_i$ and $(v_{R,i})_i$ are weakly increasing on L and R, correspondingly.*

The proof is by induction on i.

Example. In Figure 1 the path from point p_3 on consists of two Case 2 segments with associated points $(v_{L,4}, v_{R,4})$ and $(v_{L,5}, v_{R,4})$. The endpoint of the second segment is determined by event no. 2.

Summarizing, we obtain the following.

Theorem 2.9 *Let P be a street consisting of n edges. Then algorithm* HighLevelStrategy *finds a path w from s to g in P that consists of $m = O(n)$ line segments. If $l_i l_{i+1} \ldots l_j$ is a sequence of Case 2 segments of w followed by a sequence $l_{j+1} l_{j+2} \ldots l_k$ of Case 1 segments then for each point z in $l_i l_{i+1} \ldots l_j$ the first vertex visited by the shortest path from z to g is p_{j+1}, and $l_{j+2} \ldots l_k$ is a piece of the shortest path from s to g.*

Corollary 2.10 *The memory size needed by the robot does not depend on the complexity of the street but only of the maximum complexity of the visibility polygons encountered.*

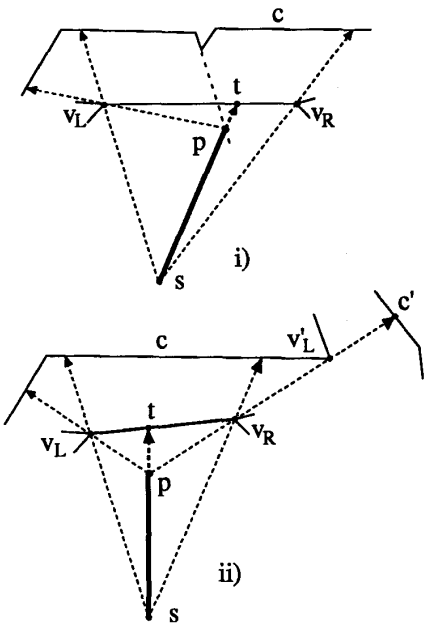

Figure 8: i) Part of c may become invisible. ii) Event no. 3.

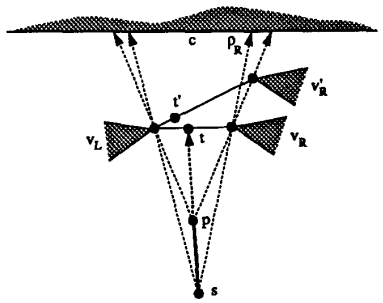

Figure 9: Event no. 4.

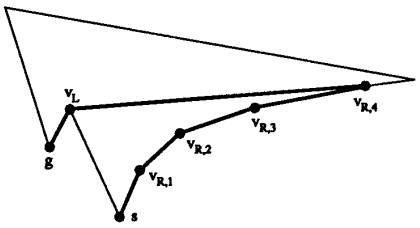

Figure 10: To walk towards the closer candidate corner can result in an unbounded detour.

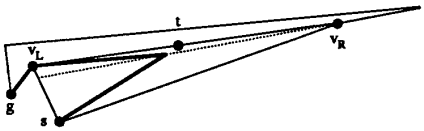

Figure 11: To head for the middle of $\overline{v_L v_R}$ can result in an unbounded detour, too.

3 Minimizing the Local Absolute Detour

The problem not settled by the high-level strategy is *how* to choose the target point, t in $\overline{v_L v_R}$, in Case 2; see Figure 9.

An obvious idea is to choose the closer one of v_L and v_R. However, by this strategy the robot can be lured off the shortest path arbitrarily far, see Figure 10.

Another obvoius approach could be to head for the point in the middle of $\overline{v_L v_R}$. But in general, this strategy does not work either, as Figure 11 shows.

The strategy we propose tries to minimize the *local absolute detour*. Suppose the robot is for the first time in the situation of Case 2, as shown in Figure 9. At the latest upon arriving at $t \in \overline{v_L v_R}$ an event will occur. If v_L turns out to be the correct corner then the robot has to walk from t to v_L, causing the absolute detour $D_L(t) = st + tv_L - sv_L$, where vw denotes the distance between the points v and w. Otherwise, it must walk to v_R, resulting in the detour $D_R(t) = st + tv_R - sv_R$.

Lemma 3.1 *The maximum of $D_L(t)$ and $D_R(t)$ becomes minimal iff t is chosen such that*

$$v_L t = \frac{sv_L - sv_R + v_L v_R}{2}.$$

Proof: An application of the law of cosine shows that the function $D_L(t)$ is strictly increasing from 0 to a value greater than 0 as t moves from v_L to v_R; similarly, $D_R(t)$ is strictly decreasing from a positive value to 0. Thus, the maximum of both becomes minimal at the unique point t where the values are equal. □

In Figure 9 the robot chooses t by the above formula and starts walking towards t. On arriving at p', the robot sees vertex v'_R and chooses its next target point t' on $\overline{v_L, v'_R}$ by the same rationale. But this time the length of the shortest path from s to v'_R, $sv_R + v_R v'_R$, is taken into account, so t' is determined by

$$v_L t' = \frac{sv_L - (sv_R + v_R v'_R) + v_L v'_R}{2}.$$

Generally, the low-level strategy *lad* chooses the next target point t_{k+1} in $\overline{v_{L,k+1} v_{R,k+1}}$ according to the formula

$$v_{L,k+1} t_{k+1} := \frac{A_{k+1} - B_{k+1} + v_{L,k+1} v_{R,k+1}}{2}.$$

The point t_{k+1} minimizes the maximum of the possible absolute detours

$$D_L(t) := \sum_{j=i}^{k} p_{j-1}p_j + p_k t + t v_{L,k+1} - A_{k+1}$$

$$D_R(t) := \sum_{j=i}^{k} p_{j-1}p_j + p_k t + t v_{R,k+1} - B_{k+1}$$

where t ranges in $\overline{v_{L,k+1} v_{R,k+1}}$.

The performance of strategy *lad* depends only on its behavior in convex funnels, since these are the only ambiguous situations where the robot does not follow the shortest path, due to Theorem 2.9.

In practice, strategy *lad* works very well. Though we have deliberately tried to create bad funnels, we have not been able to construct a funnel whose relative detour exceeds $D = 1.8$. This is complemented by the following lower bound.

Theorem 3.2 *For each possible strategy S*

$$\inf D_S(P) \geq \sqrt{2} = 1.414\ldots$$

holds, the infimum being taken over all streets P.

Proof: Let P denote the polygon depicted in Figure 12. P is not a complete street since the goal has not yet been specified. The robot cannot look into the caves before it reaches the dotted line, b. Suppose that its first point of contact lies to the right of h, depending on strategy S. In this case, the goal is put

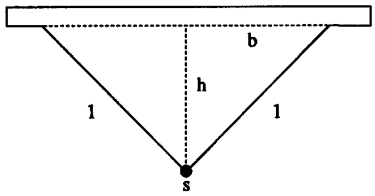

Figure 12: Establishing a lower bound for the relative detour.

into the left cave. Then the total length of the robot's path from s to g is as least as large as

$$\text{length of } h + \frac{1}{2} \text{ length of } b = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} = \sqrt{2}$$

whereas the shortest path from s to g is of length 1. □

One might object that a street whose "breadth" exceeds its "length" makes a poor example. But rather than placing the goal in one of the caves, we could as well glue on another copy of P, and iterate the construction. This would lead to a street of unbounded length and bounded breadth.

4 An Upper Bound for the Global Relative Detour

Let (C_L, C_R) be a funnel as depicted in Figure 13. Since it is difficult to estimate the length of the path generated by strategy lad directly, we prove a bound for the *longer* path shown in Figure 13 that results if the robot does not react to the new visibility information obtained at point p_i but continues walking to its target point, t_i.

Theorem 4.1 *With the above notations the following holds.*

$$p_0t_1 + \sum_{j=2}^{n} t_{j-1}t_j + t_nv_n \leq (1 + \frac{3}{2}\pi)A_n.$$

A fortiori, this yields an upper bound for the relative detour caused by strategy *lad.*

Theorem 4.2 *For each street P*

$$D_{lad}(P) \leq 1 + \frac{3}{2}\pi = 5.71\ldots.$$

In order to prove Theorem 4.1 we have to estimate the length of the path depicted in Figure 13 against the length of the left convex chain, C_L. Clearly, the

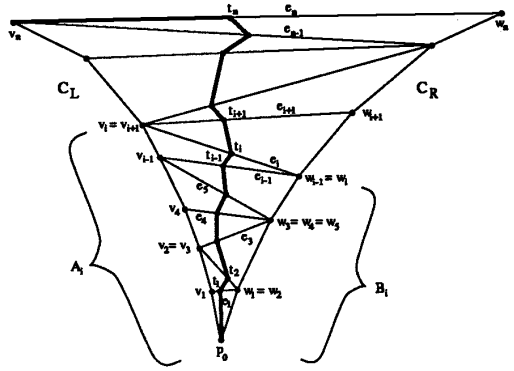

Figure 13: Point t_i is determined by $v_it_i := \frac{A_i - B_i + v_iw_i}{2}$.

path becomes only longer if we insert additional edges e_i that do not violate the above conditions. Thus, we may assume that each vertex of the funnel is the endpoint of an edge e_i.

The edges split the funnel into two types of triangles. If $w_{i-1} = w_i$ holds for two consecutive edges e_i, e_{i-1}, then the included triangle T_{i-1} shares its third side with C_L. If $v_i = v_{i+1}$ then the included triangle T_i has its third side on C_R. The bottommost triangle of the funnel is special; we define it to be of the former type by putting $v_0 := p_0$, $w_0 := w_1$, and $t_0 := p_0$.

The following lemma shows that the target points can be computed incrementally.

Lemma 4.3 *Assume that $v_i = v_{i+1}$ holds. Then*

1. $v_it_{i+1} = \frac{1}{2}(v_it_i - t_iw_i - w_iw_{i+1} + v_iw_{i+1})$,
2. $t_{i+1}w_{i+1} = \frac{1}{2}(t_iw_i + w_iw_{i+1} - v_it_i + v_iw_{i+1})$,
3. $v_it_{i+1} \leq v_it_i$.

Symmetric formulae hold in the case $w_{i-1} = w_i$.

Next, we distribute the length of the path to be estimated among the triangles T_i.

Definition 4.4 Let e_i, $i < n$, be an edge in the funnel. Then the cost of the triangle T_i above e_i is defined by

$$cost\ (T_i) := t_it_{i+1} + v_{i+1}t_{i+1} - v_it_i.$$

Note that for the special triangle T_0 we obtain $cost\ (T_0) = p_0t_1 + v_1t_1$, since $t_0 = p_0 = v_0$. Clearly,

the sum of these quantities telescopes into the length of the path,

$$\sum_{i=0}^{n-1} cost\,(T_i) = p_0 t_1 + \sum_{j=2}^{n} t_{j-1} t_j + t_n v_n.$$

The following lemma provides the main tool for estimating the cost contribution of a single triangle. We omit the lenghty proof.

Lemma 4.5 *Let α_i denote the angle between e_i and e_{i+1}.*

1. *If $v_i = v_{i+1}$ then $cost\,(T_i) \leq v_i t_i \sin\alpha_i$*
2. *If $w_i = w_{i+1}$ then $cost\,(T_i) \leq t_i w_i \sin\alpha_i + v_i v_{i+1}$*

The triangles in the funnel can be grouped into *left fans*, denoting maximal sequences of triangles that share but a vertex with C_L, and *right fans*. By convention, the bottommost triangle, T_0, is (part of) a right fan. We define the *cost of a fan* to be the sum of the costs of its constituing triangles.

Lemma 4.6 *Let $F = T_i T_{i+1} \ldots T_k$ be a fan in the funnel.*

1. *If F is a left fan then*

$$cost\,(F) \leq v_i t_i \sum_{j=i}^{k} \alpha_j =: Bcost\,(F)$$

2. *If F is a right fan then*

$$cost\,(F) \leq t_i w_i \sum_{j=i}^{k} \alpha_j + \sum_{j=i}^{k} v_j v_{j+1} =: Bcost\,(F)$$

Finally, we need the following technical result.

Lemma 4.7 *Let a, b, and c be the sides of a triangle, let ρ denote the angle opposite to a, and let A be a curve connecting the endpoints of a. Then the following holds for the lengths of these pieces.*

$$c\rho + \frac{1}{2}(A - c + b)\pi \leq \frac{3}{2} A\pi.$$

Now we can prove Theorem 4.1.

Proof: (Sketch) We show by induction on the number of left fans in the funnel that

$$\sum_{F \text{ fan}} Bcost\,(F) \leq (1 + \frac{3}{2}) A_n$$

holds, where *Bcost* is as defined in Lemma 4.6.

A funnel without a left fan consists of a single right fan F that includes the bottommost triangle. We apply Lemma 4.7 to the triangle (v_0, w_0, v_n). The same works if the funnel ends with a right fan. If it ends with a left fan, F, then the *Bcosts* of F and of the right fan, F', below F can only patially be charged to the last piece of the left chain bordering F'. The rest, r, is charged to the funnel Q' that results from the original one by cutting off F and F'. More precisely, if the left fan on top of Q' is enlarged by drawing the tangent from the endpoint of the shortened left chain to the right chain, then r does not exceed the *Bcost* of the left subfan newly added. Since the resulting funnel has one fan less than the original one, the induction hypothesis can be applied. □

5 Concluding Remarks

We have introduced a class of simple polygons in order to describe streets of varying breadth that may contain many curves but no crossings. Without knowing the street in advance, a mobile robot equipped with an on-board vision system can find a path from the start to the goal large portions of which are part of the shortest path. There are, however, situations where the robot cannot know if the street ahead is turning left or right; then a deviation from the shortest path is unavoidable.

In order to keep the deviation short, we have designed a strategy that tries to minimize the local absolute detour and thus guarantees the overall relative detour to be bounded.

One challange is to close the gaps between the proven upper bound of 5.72, the empirical upper bound of 1.8, and the lower bound of 1.4. Though it seems reasonable to study a continuous model (with curves instead of polygonal chains) it is not clear if the theory of differential equations can help.

Another question addresses different low-level strategies. One alternative is strategy *spl* that always follow the shortest path to the line segment $\overline{v_L v_R}$. Despite being simpler than *lad*, strategy *spl* is still difficult to analyze. Also, one can construct examples where the detour caused by *spl* exceeds 1.8, our empirical bound for *lad*. This approach is currently under investigation.

A further problem concerns the generalization to a kinodynamic model, where the robot has a unit mass and is, in each direction, capable of a maximal velocity and acceleration. Here no longer a short path is asked for, but a fast trajectory that has to be safe in the sense that the robot can always stay on the street no

matter what the next curve looks like. An additional challenge arises if the robot's speed is so large that the time needed for image processing and for deciding about the next action must be taken into account—a situation well known from real life.

References

[1] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. Tech. Rep. No. RC 16452 (#73101) 1/17/91, Computer Science, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, USA, 1991. Also in *Proc. ACM STOC*, 1990.

[2] B. Chazelle. Efficient polygon triangulation. In *Proc. 31st IEEE FOCS*, 1990.

[3] A. Datta and K. Krithivasan. Path planning with local information. In *Proc. Foundations of Software Technology and Theoretical Computer Science*, New Delhi, India, 1988.

[4] X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment. In *Proc. 32nd IEEE FOCS*, 1991.

[5] P. Eades, X. Lin, and N. C. Wormald. Performance guarantees for motion planning with temporal uncertainty. Tech. Rep. No. 173, Key Center for Software Technology, Dept. of Computer Sc., The University of Queensland, Australia, 1990.

[6] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. In *Proc. 3rd ACM Symposium on Computational Geometry*, pages 50–63, Waterloo, 1987.

[7] Ch. Icking and R. Klein. The two guards problem. In *Proc. 7th ACM Symposium on Computational Geometry*, North Conway, 1991.

[8] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, 6(4):462–472, 1990.

[9] V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[10] J. S. B. Mitchell. An autonomous vehicle navigation algorithm. In *Proc. SPIE Applications of Artificial Intelligence*, 485:153–158, 1984.

[11] J. S. B. Mitchell. Algorithmic approaches to optimal route planning. In *Proc. SPIE Conference on Mobile Robots*, 1990.

[12] C. H. Papadimitriou and M. Yanakakis. Shortest paths without a map. In *Proc. 16th ICALP* , 610–620, 1989.

[13] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. Tech. Rep. B 90-07, FU Berlin, FB Mathematik, Serie B Informatik, 1990.