RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN
INSTITUT FÜR INFORMATIK I

Elmar Langetepe

Online Motion Planning

MA INF 1314

Summersemester 2016

Manuscript: Elmar Langetepe

We would like to guarantee a correct detection for tha maximal error and deviation. The correct angle of a convex vertex is $\frac{3}{2}\pi$ and the correct angle of a reflex vertex is $\frac{\pi}{2}$. Therefore we require:

$$\begin{array}{lcl} \frac{3}{2}\pi - 2\delta - \rho > \pi & \text{und} & \frac{\pi}{2} + 2\delta + \rho < \pi \\ \Leftrightarrow 2\delta + \rho < \frac{\pi}{2} & & \Leftrightarrow 2\delta + \rho < \frac{\pi}{2}\,. \end{array}$$

Additionally, we would like to distinguish between horizontal and vertical edges after hitting an edge. Either we slip along the vertical edge or we start at the horizontal with the simple counter. Again we assume the worst-case situation; see Figure 2.9.

We measure the turning angle γ for the corresponding edge. For a horizontal edges this is exact $-\frac{\pi}{2}$; see Figure 2.9(i). If this angle is between $-\frac{\pi}{4}$ and $-\frac{3\pi}{4}$ we conclude that we have a horizontal edge. Otherwise the edge is assumed to be vertical. Note that γ is always negative. We assume that the deviation from the starting direction is φ.

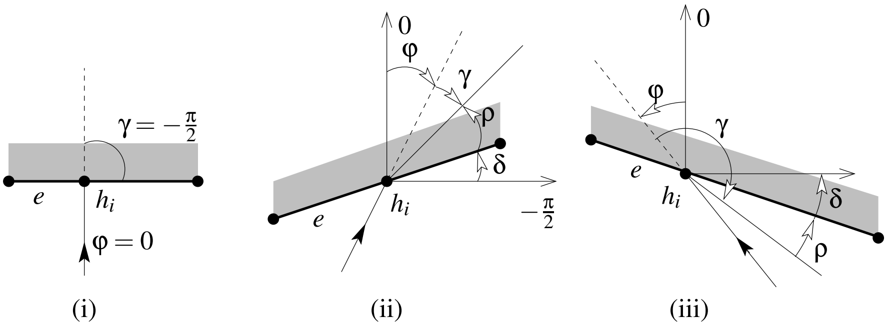

Figure 2.9: Hitting a horizonal edges (i) in the error-free case, (ii) for small absolut γ, (iii) for large absolut γ.

In Figure 2.9(ii) the deviations φ, δ and ρ should make $|\gamma|$ as small as possible and still smaller than $-\frac{\pi}{4}$, φ is negative. In Figure 2.9(iii) the deviations φ, δ and ρ should make $|\gamma|$ as large as possible and larger than $-\frac{3\pi}{4}$, φ is positive. We conclude from Figure 2.9(ii)

$$\gamma = -\frac{\pi}{2} - \varphi + \delta + \rho < -\frac{\pi}{4} \;\Leftrightarrow\; -\frac{\pi}{4} + \delta + \rho < \varphi\,,$$

and from Figure 2.9(iii)

$$\gamma = -\left(\frac{\pi}{2} + \varphi + \delta + \rho\right) > -\frac{3}{4}\pi \;\Leftrightarrow\; \frac{\pi}{4} - \delta - \rho > \varphi\,.$$

We We detect horizontal edges precisely if $\varphi(h_i) \in \left] -\frac{\pi}{4} + \delta + \rho, \frac{\pi}{4} - \delta - \rho \right[$ holds. Therefore we require $\delta + \rho < \frac{\pi}{4}$. A maximal deviation of $\frac{\pi}{4} - \delta - \rho$ would be enough for correct detections. Since we might start the free space move with an error of δ at a vertex we require $\frac{\pi}{4} - 2\delta - \rho$ for the deviation. □

Exercise 15 *In the above corollary we can set $\delta = 0$ and $\rho = 0$ and require that we do not deviate in the free space by an angle of $\pi/4$. Why is this different to the error-free case where an error of less than $\pi/2$ was allowed for the free space movements.*

2.2 Navigation with touching sensor

We distinguish between the term Navigation for visiting a given target (known coordinates) and the term Searching for searching for an unknown goal (unknown coordinates). The family of the so-called Bug-Algorithms are the first algorithms for the navigation task in polygonal environments[2]. The first

[2] In this case Bug is not meant as a synonym for an error.

simple strategies have been introduced by Lumelsky and Stepanov [LS87], extensions and modifications came from Sankaranarayanan et al. [SM92, SV90a, SV90b, SV91]. Many variants have been discussed since then. Bug-variants have been practically used for the navigation of some of the Mars rovers like Sojourner or Bridget, (see also RoverBug, [LB99]).

In the following we assume that the coordinates of the target are known and that the agent have a finite storage so that coordinates of points and /or length of (sub)path can be stored. The agent also is aware of the coordinates of its current position, for example by GPS.

Any Bug-algorithm runs with the same principle and actions: The agent moves toward the target until an obstacle is visited (Move-To-Target action) Then the agent follows the wall of the obstacle for a while (Follow-Wall action) until some condition triggers the next movement in the free space toward the target. The leaving condition is the main difference between the Bug-variants.

We assume that the agent R is point-shaped and equipped with a touch sensor for the Follow-Wall action. We make use of the following notations:

- $|pq|$ denotes the distance between two points p and q,
- $D := |st|$ denotes distance from start s to target t,
- π_S denotes the path of a strategy S from s to t; $|\pi_S|$ denotes the length of this path where $|\pi_S| := \infty$, if there is no such path,
- $\mathrm{U}(P_i)$ denotes the perimeter of the obstacle P_i.

2.2.1 Strategies of Lumelsky and Stepanov

The first algorithm Algorithm 2.2, Bug1, leaves an obstacle P_j at a point $p \in P_i$ that is the closest point to the target. This defines a sequence of **Hit-Points** h_i, where the agent hits an obstacle and **Leave-Points** ℓ_i, where the agent leaves an obstacle. Since the coordinates of the target and the coordinates of the current position are known, the agent can calculate the corresponding distances. Additionally, by successively counting small steps, the agent can calculate the path length of the path along the boundary during the circumnavigation and also the path length to the currently computed optional leave-point. Additionally, the path length (along the boundary) to the With these values the agent can perform step 3 of Algorithm 2.2 Figure 2.10 shows an example for the path of Bug1.

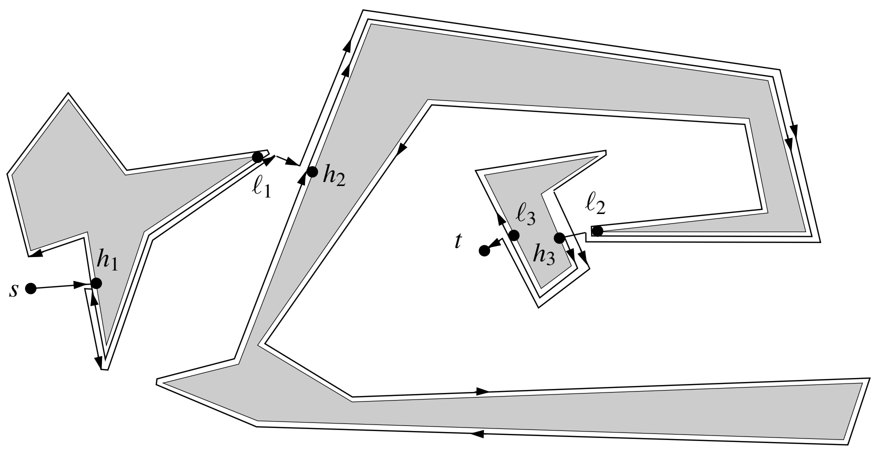

Figure 2.10: Example execution of strategy Bug1.

We assume that there is a finite number of polygonal obstacles and that the obstacles do not touch or intesect. The number of polygons is finite in the sense that any circle of fixed radius contains only finitely many *obstacles* P_i.

Algorithm 2.2 Bug1

0. $\ell_0 := s$, $i := 1$
1. From ℓ_{i-1} move toward the target, until
   (a) Target is visited: Stop!
   (b) An obstacle is reached at point h_i. If $h_i = \ell_{i-1}$: The goal cannot be reached.
2. Surround the obstacle in cw-order — keep track of the point ℓ_i on the boundary with the shortest distance to t —, until
   (a) Target is visited: Stop!
   (b) h_i is reached.
3. Move along the shortest path along the boundary to ℓ_i.
4. Increase i, GOTO 1.

Theorem 2.13 *(Lumelsky, Stepanov, 1985)*
Strategy Bug1 finds a path from a starting point s to a target t, if such a path exists. *[LS87]*

Proof. For the sequence of hit- und leave-points we have

$$|st| \geq |h_1 t| \geq |\ell_1 t| \ldots \geq |h_k t| \geq |\ell_k t|.$$

Since for any visited obstacles we choose a leave-point that is closest to the target, any obstacles can be left. Otherwise, if this is not the case, the obstacle would fully enclose the target. This also means that we have a strict $>$ in the above sequence. Any obstacle can be visited only once and the finite number of obstacles within the circle of radius $|st|$ around t lead to a finite sequence which ends at the target. □

For the performance we conclude:

Theorem 2.14 *(Lumelsky, Stepanov, 1985)*
Let π_{Bug1} denote the path from s to t, for the successful application of the strategy Bug1. *[LS87] We have:*

$$|\pi_{\text{Bug1}}| \leq D + \frac{3}{2}\sum_i \mathrm{U}(P_i).$$

Proof. We subdivide the path into the movements along the boundary of the obstacles P_i and the movements in the free space. Since step 3. of Algorithm 2.2 makes use of a shortest path we have path length $\frac{3}{2}\sum \mathrm{U}(P_i)$ for any visited obstacle. It remains to calculate the length D' for the free space movements. We show that $D' \leq D$ holds.

$$\begin{aligned}
D' &= |sh_1| + |\ell_1 h_2| + \ldots + |\ell_{k-1} h_k| + |\ell_k t| \\
&\leq |sh_1| + |\ell_1 h_2| + \ldots + |\ell_{k-1} h_k| + |h_k t| \\
&= |sh_1| + |\ell_1 h_2| + \ldots + |\ell_{k-1} t| \\
&\ldots \\
&\leq |sh_1| + |\ell_1 h_2| + |h_2 t| \\
&= |sh_1| + |\ell_1 t| \\
&\leq |sh_1| + |h_1 t| = |st| = D
\end{aligned}$$

We can compare the above result with the lower bound Theorem **??** and conclude that in comparison to any other Bug-strategy the strategy Bug1 can be consodered to be $\frac{3}{2}$-competitive. □

Corollary 2.15 *Bug1 is $\frac{3}{2}$-competitive in comparison to arbitrary Bug-like online strategyies.*

In the next variant we would like to avoid complete circumnavigations of the obstacles. Therefore we make use of a line G passing through the segment st. At any time during the Wall-Follow action we will try to move toward the target if we reach a point at G that is closer to t than the previous hit-point; see Algorithm 2.3. Note that by this action, it is possible to visit an obstacles more than once which was impossible for Bug1. h_j and ℓ_j do no longer denote hit- and leave-points of the j-th obstacle.

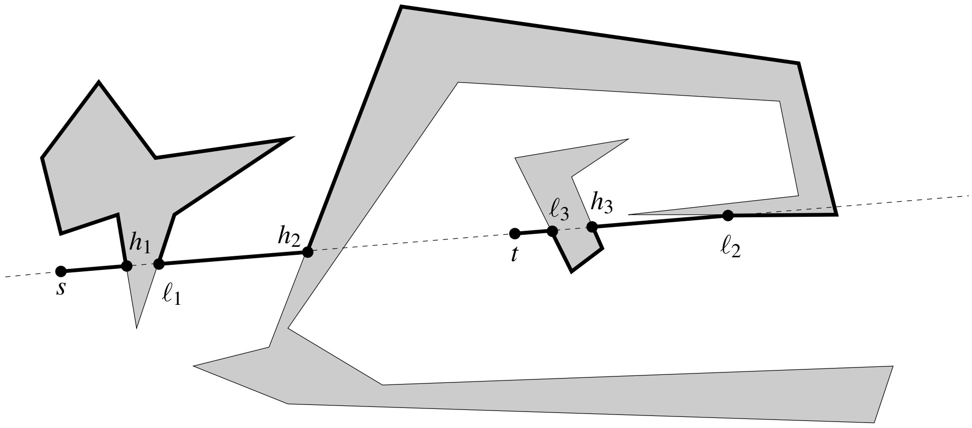

Figure 2.11: Example of the execution of the strategy Bug2.

Figure 2.12 shows an example, where an obtacle is visited more than once. After hit-point h_3 the agent does not leave the obstacle at p_1 or ℓ_1 since $|h_3t|$ is smaller thanthe distance of p_1 and ℓ_1 to t. At p_2 and p_3 the agent cannot leave the obstacle since the segments $\overline{p_{2/3}t}$ are blocked by the obstacle.

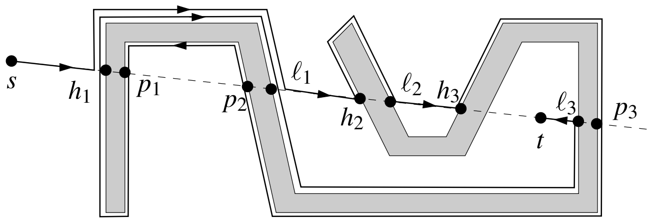

Figure 2.12: The execution of Bug2 can lead to several visits of the same obstacle.

The number of polygons is finite in the sense that any circle of fixed radius contains only finitely many *obstacles P_i*.

Lemma 2.16 *The strategy Bug2 meets finitely many obstacles.*

Proof. In step 2b of Algorithm 2.3 the agent leaves an obstacle only if $|\ell_jt| < |h_jt|$ holds. Since the circle of radius $|\overline{st}|$ around t contains only finitely many obstacles we can hit only finitely many obstacles. □

The number of surroundings depend on the intersections of the line passing through st with the boundary of the relevant obstacles.

Lemma 2.17 *Let n_i denote the number of intersections of the line $\overleftrightarrow{st}$ passing through st with the boundary of polygon P_i. The strategy Bug2 visits any point of P_i only $\frac{n_i}{2}$ times.*

Proof.

Bug2 successively defines pairs (h_j, ℓ_j) of hit- und leave-points and by the leave condition we have

$$|h_jt| > |\ell_jt| > |h_{j+1}t|.$$

Algorithm 2.3 Bug2

0. $\ell_0 := s$, $j := 1$

1. From ℓ_{j-1} move toward the target, until

   (a) Target is reached: Stop!

   (b) An obstacle is visited at h_j.

2. Surround the obstacle in cw-order, until

   (a) Target is reached: Stop!

   (b) The line passing segment st is visited at point q, $|qt| < |h_j t|$ and $\overline{qt}$ is *free*, such that we can leave the obstacle from q toward the target.
   Set $\ell_j := q$, $j := j+1$ and GOTO 1.

   (c) h_j is visited again without reaching a point q as in described in b). The target cannot be reached. erreicht werden.

This means that any point is only once a leave-point or a hit-point and any intersection point can appear only in one pair (h_j, ℓ_j). On the other hand a single pair can only lead to one full surrounding, if the same hit-point is visited, the strategy stops. We have at most $\frac{n_i}{2}$ pairs and surroundings. □

Finally we conclude that we have only finitely many relevant intersections and either the strategy visits a current hit-point again and the corresponding obstacle enlcoses the target or we will finally succeed.

Corollary 2.18 *Strategy Bug2 is successful, if the target can be reached.*

The performance of Bug2 is given in the following statement.

Theorem 2.19 *(Lumelsky, Stepanov, 1985)*
Let π_{Bug2} denote the path from s to t, for the successful application of strategy Bug2. We have

$$|\pi_{\mathrm{Bug2}}| \leq D + \sum_i \frac{n_i \, \mathrm{U}(P_i)}{2}.$$

Here P_i is an obstacle that is visited during the execution of Bug2. *[LS87]*

Proof. The term $\sum \frac{n_i \mathrm{U}(P_i)}{2}$ follows from Lemma 2.17. For the length of the free space movements, say D', between the obstacles, we make use of the same arguments as in the proof of Theorem 2.14 and conclude $D' \leq D$. □

Bug2 is not always better than Bug1. Obviuously, in the presence of convex obstacles, Bug2 outperforms Bug1.

Corollary 2.20 *For a polygonal scene with convex obstacles the successful application of strategy Bug2 has path length*

$$|\pi_{\mathrm{Bug2}}| \leq D + \sum_i \mathrm{U}(P_i).$$

Exercise 16 *Compare the variants Bug1 and Bug2. Present an example where Bug1 outperforms Bug2. Show that for both strategies the performance guarantee is tight.*

2.2.2 Strategies from Sankaranarayanan and Vidyasagar

Many variants of the Bug-strategies have been discussed. Many of them make use of more sensor power for local improvement. For example a VisBug2 strategy makes use of a visibility range and can find local short-cuts for the Bug2 path. We would like to mention some structural different variants from Sankaranarayanan and Vidyasagar. The reason is that we would like to show that some local optimization can have unexpected disadvatages.

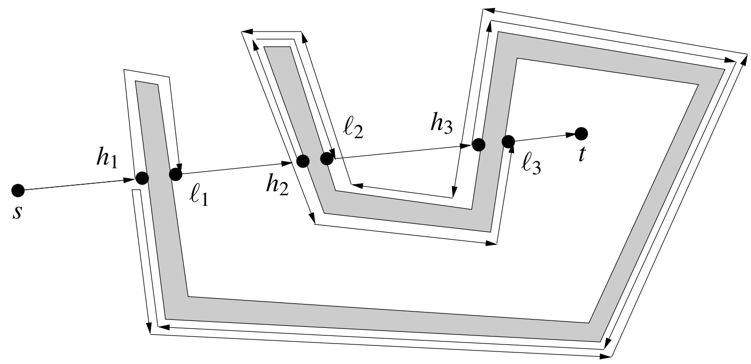

Figure 2.13: Example of the execution of Change1.

Bug1 fully surrounds any obstacle, Bug2 tries to avoid this by moving toward the goal a bit earlier. In this case a single obstacle can be visited many times. Algorithm 2.4 tries to avoid this behaviour: If a surrounding is started, and an old hit- or leave-point (not the current hit-point!) is visited, the strategy starts moving along the boundary in ccw-order; see Figure 2.13.

Theorem 2.21 *(Sankaranarayanan, Vidyasagar, 1990)*
For the length of the path of the successful application of strategy Chang1 we have *[SV90a]*

$$|\pi_{\text{Change1}}| \leq D + 2 \cdot \sum_i \mathrm{U}(P_i).$$

Proof. Exercise □

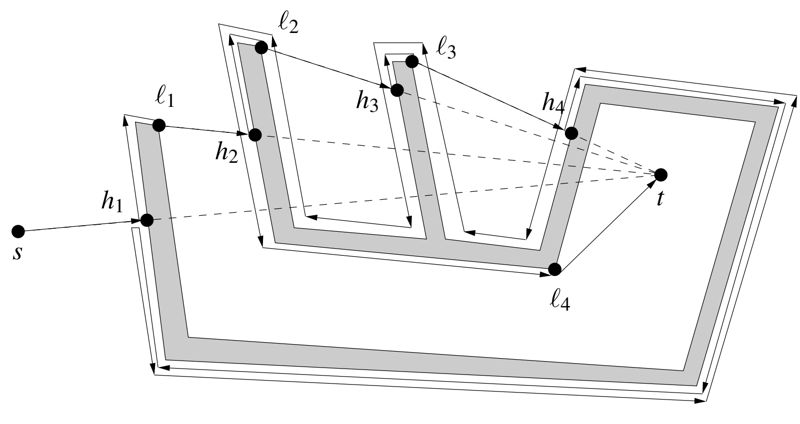

Figure 2.14: Example execution of strategy Change2.

Strategy Change2 (Algorithm 2.5) differs from Change1 only in the leaving condition. The leave-point is not restricted to a point on the line $\overleftrightarrow{st}$. As soon as there is a point q on the boundary in the Follow-Wall action that is closer to the target than the distance $|ht|$ for the last hit-point,we will leave the obstacle toward the target, if this is possible. Note that such a behaviour can also be used for a variant of Bug2.

Theorem 2.22 *(Sankaranarayanan, Vidyasagar, 1990)*
For the length of the path of the successful application of strategy Chang1 we have *[SV90b]*

$$|\pi_{\text{Change2}}| \leq D + 2 \cdot \sum_i \mathrm{U}(P_i).$$

Proof. Exercise □

Exercise 17 *Present proofs for the above two Theorems. Show that the bounds are tight.*

Algorithm 2.4 Wechsel1

0. $\ell_0 := s$, $i := 1$
1. Move from ℓ_{i-1} along the line passing s and t toward the target, until
   (a) Target is reached: Stop!
   (b) An obstacle is reached at h_i.
2. Surround the obstacle, until
   (a) Target is reached: Stop!
   (b) The line passing s and t is visited a some point q such that the distance from q to t is smaller than $h_i t$ and the segment $\overline{qt}$ is *free* (see refalgobug2).
   Set $\ell_i := q$, $i := i+1$ und GOTO 1.
   (c) A hit- or leave-point h_j orℓ_j with $j < i$ is visited: Move back to h_i in ccw-order and start ccw-order surrounding under condition (a), (b) oder (d) (not (c) again!)
   (d) h_i is visited again without reaching a point as indicated in (b) or (c). The goal is enclosed by an obstacle.

Algorithm 2.5 Wechsel2

As Change1, but:

0. $\ell_0 := s$, $i := 1$
1. Move from ℓ_{i-1} along the line passing s and t toward the target, until

2.(b) A point q is visited such that the distance from q to t is smaller than $h_i t$ and the segment $\overline{qt}$ is *free* (see refalgobug2).
Set $\ell_i := q$, $i := i+1$ und GOTO 1.

Bibliography

[Ad80] H. Abelson and A. A. diSessa. *Turtle Geometry*. MIT Press, Cambridge, 1980.

[AFM00] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Comput. Geom. Theory Appl.*, 17:25–50, 2000.

[AKS02] Susanne Albers, Klaus Kursawe, and Sven Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32:123–143, 2002.

[BRS94] Margrit Betke, Ronald L. Rivest, and Mona Singh. Piecemeal learning of an unknown environment. Technical Report A.I. Memo No. 1474, Massachusetts Institute of Technology, March 1994.

[DJMW91] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *Transactions on Robotics and Automation*, 7:859–865, 1991.

[DKK01] Christian A. Duncan, Stephen G. Kobourov, and V. S. Anil Kumar. Optimal constrained graph exploration. In *Proc. 12th ACM-SIAM Symp. Discr. Algo.*, pages 307–314, 2001.

[DKK06] Christian A. Duncan, Stephen G. Kobourov, and V. S. Anil Kumar. Optimal constrained graph exploration. *ACM Trans. Algor.*, 2:380–402, 2006.

[GR03] Yoav Gabriely and Elon Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comput. Geom. Theory Appl.*, 24:197–224, 2003.

[IKKL00a] Christian Icking, Thomas Kamphans, Rolf Klein, and Elmar Langetepe. Exploring an unknown cellular environment. In *Abstracts 16th European Workshop Comput. Geom.*, pages 140–143. Ben-Gurion University of the Negev, 2000.

[IKKL00b] Christian Icking, Thomas Kamphans, Rolf Klein, and Elmar Langetepe. Exploring an unknown cellular environment. Unpublished Manuscript, FernUniversität Hagen, 2000.

[IKKL05] Christian Icking, Tom Kamphans, Rolf Klein, and Elmar Langetepe. Exploring simple grid polygons. In *11th Internat. Comput. Combin. Conf.*, volume 3595 of *Lecture Notes Comput. Sci.*, pages 524–533. Springer, 2005.

[IPS82] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686, 1982.

[KL03] Tom Kamphans and Elmar Langetepe. The Pledge algorithm reconsidered under errors in sensors and motion. In *Proc. of the 1th Workshop on Approximation and Online Algorithms*, volume 2909 of *Lecture Notes Comput. Sci.*, pages 165–178. Springer, 2003.

[LB99] Sharon Laubach and Joel Burdick. RoverBug: Long range navigation for mars rovers. In Peter Corke and James Trevelyan, editors, *Proc. 6th Int. Symp. Experimental Robotics*, volume 250 of *Lecture Notes in Control and Information Sciences*, pages 339–348. Springer, 1999.

[Lee61] C. Y. Lee. An algorithm for path connections and its application. *IRE Trans. on Electronic Computers*, EC-10:346–365, 1961.

[LS87] 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.

[Sha52] Claude E. Shannon. Presentation of a maze solving machine. In H. von Foerster, M. Mead, and H. L. Teuber, editors, *Cybernetics: Circular, Causal and Feedback Mechanisms in Biological and Social Systems, Transactions Eighth Conference, 1951*, pages 169–181, New York, 1952. Josiah Macy Jr. Foundation. Reprint in [Sha93].

[Sha93] Claude E. Shannon. Presentation of a maze solving machine. In Neil J. A. Sloane and Aaron D. Wyner, editors, *Claude Shannon: Collected Papers*, volume PC-03319. IEEE Press, 1993.

[SM92] A. Sankaranarayanan and I. Masuda. A new algorithm for robot curvefollowing amidst unknown obstacles, and a generalization of maze-searching. In *Proc. 1992 IEEE Internat. Conf. on Robotics and Automation*, pages 2487–2494, 1992.

[Sut69] Ivan E. Sutherland. A method for solving arbitrary wall mazes by computer. *IEEE Trans. on Computers*, 18(12):1092–1097, 1969.

[SV90a] A. Sankaranarayanan and M. Vidyasagar. A new path planning algorithm for a point object amidst unknown obstacles in a plane. In *Proc. 1990 IEEE Internat. Conf. on Robotics and Automation*, pages 1930–1936, 1990.

[SV90b] A. Sankaranarayanan and M. Vidyasagar. Path planning for moving a point object amidst unknown obstacles in a plane: A new algorithm and a general theory for algorithm developments. In *Proceedings of 1990 IEEE Conf. on Decision and Control*, pages 1111–1119, 1990.

[SV91] A. Sankaranarayanan and M. Vidyasagar. Path planning for moving a point object amidst unknown obstacles in a plane: The universal lower bound on the worst case path lengths and a classification of algorithms. In *Proc. 1991 IEEE Internat. Conf. on Robotics and Automation*, pages 1734–1741, 1991.

Index