

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN
INSTITUT FÜR INFORMATIK I



Elmar Langetepe
Online Motion Planning

MA INF 1314

Sommersemester 2016
Manuscript: Elmar Langetepe

Chapter 2

Polygonal environments

In this section we turn over to planar environments modelled by (a set of) simple polygons. We assume that a set of simple polygons P_i for $i = 1, \dots, k$ is given. Two polygons do neither intersect nor touch each other. The number of polygons is finite in the sense that any circle of fixed radius contains only finitely many *obstacles* P_i .

In the following sections an agent tries to escape from a labyrinth (modelled by polygons) or tries to find a target t in a polygonal environment. We assume that the agent is point-shaped and thus consider curves in the plane. From a practical point of view one might think that a physical robot somehow *follows* the corresponding curve without actually running precisely on it. Additionally, in the following configurations, the agent has only a limited storage.

We make use of the following conventions. If the coordinates of the target are given, the task of the agent is denoted as “Navigation”. On the other hand, if the coordinates are not known the task of the agent is “Searching”. We will consider different sensor models.

Some of the following algorithms – for example the Pledge-algorithm and some Bug-Variants – have been implemented as interactive Java-Applets, see

<http://web.informatik.uni-bonn.de/I/GeomLab/Polyrobot/>

2.1 Escape from the labyrinth

The task of the agent is to escape from a polygonal environment. In the geometric sense the agent escapes, if it finally hits a circle that contains all obstacles.

The agent is point-shaped and makes use of a touch-sensor for following the wall by Left-Hand-Rule or Right-Hand-Rule. Additionally, we allow that the agent can count its total turning angles in a single ; see Figure 2.2(i). The agent realizes when it hits the enclosing circle. In this case the agent successfully escaped from the labyrinth.

2.1.1 Pledge-Algorithm

Algorithm 2.1 Pledge-Algorithm

1. Choose direction φ , turn agent into direction φ .
 2. Move into direction φ , until an obstacle is met.
 3. Turn right and follow the wall by Left-Hand-Rule.
 4. Follow the wall, sum up the overall turning angles, until the angular counter gets zero, in this case GOTO (2).
-

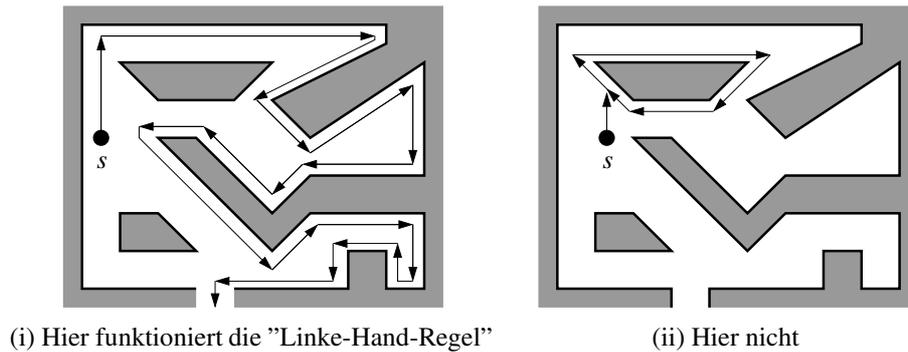
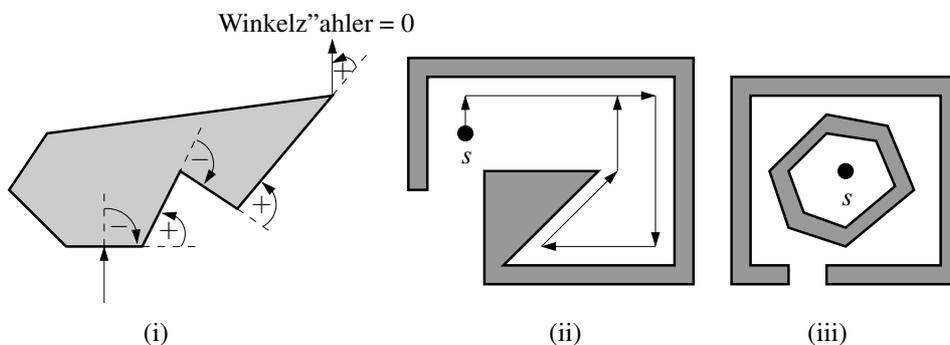


Figure 2.1: Simple strategies cannot be successful.

Note that, simple counting schemes or movements cannot succeed. For example, following the wall by Left-Hand-Rule until the agent points again into direction φ and leave the obstacle right now can result in infinite loops; see Figure 2.2(ii). Just following the boundary could let the agent stuck inside the labyrinth; see Figure 2.1.

Figure 2.2: (i) Angular counter. (ii) Leave-condition “angular counter mod $2\pi = 0$ ” is not sufficient. (iii) The agent cannot escape.

The following simple Pledge-Algorithm (Algorithm 2.1) solves the problem. For the correctness proof we require structural properties.

Lemma 2.1 *The angular counter of the Pledge-Algorithm never attains a positive value.*

Proof.

- The angular counter is initialized by zero.
 - After hitting an obstacle, the counter gets negative by the first right turn.
 - The counter changes continuously, the agent leaves the obstacle if the counter gets zero again.
- ⇒ the statement is true. □

Lemma 2.2 *If the agent does not leave the labyrinth by the Pledge-Algorithm, at the end there will be a finite path π_o , so that the agent follows this path infinitely often.*

Proof. The path generated by the Pledge-Algorithm is a polygonal chain. The set of vertices S stem from the vertex set of all obstacles plus vertices inside edges that occur when the algorithm leaves an obstacle at a vertex and hits an edges afterwards. The algorithm leaves an obstacle only at a vertex. Thus the overall number of possible vertices is bounded. If the Pledge leaves an obstacle at the same vertex twice, after that the same path π_o will be used again and again, since the algorithm is deterministic and we left the same vertex with the same counter twice.

Therefore, if such path does not exist and the agent is not successful, finally the path will not leave an obstacle anymore. Obviously, the path π_o exists in this case. \square

Lemma 2.3 *Assume that the agent does not leave the labyrinth by the Pledge-Algorithm, the above mentioned finite path π_o does not have self-intersections.*

Proof. An intersection can only occur at the boundary, since all movements (segments) in the *free space* are arranged in parallel.¹

Let us assume that a part B of π_o intersects with a former path A of π_o at the boundary of some P_i . This means that there is a point $z \in P_i$ where B (running from the free space) hits A for the first time. The corresponding segment $e \in P_i$ points into a fixed direction. Therefore after turning, the turning angle, $C_B(z')$, of B at some point $z' \in e$ closely behind z and the turning angle, $C_A(z')$, differ by $2k\pi$ for some $k \in \mathbb{Z}$. If we cause turning angle $-\beta$ for B at z , we have $C_B(z') = -\beta$ and $C_A(z') = -\beta + 2k\pi$ with $k \in \mathbb{Z}$.

For $k > 0$ we have:

$$C_A(z') = -\beta + 2k\pi > -\pi + 2\pi = \pi \quad \text{by Lemma 2.1.}$$

For $k = 0$ it is clear that $C_A(z') = C_B(z')$ and A and B will never diverge, this contradicts the given situation that A and B are parts of π_o .

From $k < 0$ we conclude $C_A(p) < C_B(p)$ for all points p between z' and the first point z'' , where the paths of A and B separate. From $C_A(p) < C_B(p)$ we conclude that at z'' the path B has $C_B(z'') = 0$ and leaves the obstacle. \square

Finally, we conclude:

Theorem 2.4 (*Abelson, diSessa*)

For any given polygonal labyrinth, the Pledge-Algorithm will be able to escape from the labyrinth from any starting point, from which a successful path exists. [Ad80]

Proof. Assume, that the agent is not successful in the given situation. By Lemma 2.2 follows the finite path π_o again and again, and the path π_o has no self-intersections. Either the agent runs along π_o in ccw-order or in cw-order. In ccw-order the angular counter will increase by $+2\pi$ in any round, which contradicts Lemma 2.1. In cw-order the angular counter will decrease by -2π in any round, thus at some point, the agent cannot leave an obstacle any more and in turn π_o already belongs to a single obstacle. The agent follows the wall of an obstacle by Left-Hand-Rule and in cw-order, this can only mean that the agent is enclosed by the obstacle; see Figure 2.2(iii). There is no way out of the labyrinth from the given starting point. \square

2.1.2 Pledge-Algorithm with sensor errors

The correctness proof of the Pledge-Algorithm (Theorem 2.4) make use of the assumptions that a point-shaped agent counts the angles without any errors and moves precisely into a specified starting direction. setzt voraus, dass der Roboter punktformig ist und fehlerfrei arbeitet. As already mentioned physically the agent need not be a point, we can assume that the agent requires some room for its movement and in principle follows a curve calculated by the Pledge-Algorithm. The main problem is that the curve itself is not computed exactly by the robots abilities. For example the agent cannot precisely measure the turning angles at the boundary and cannot precisely follow a direction.

If the agent makes gross faults, we cannot assume that the pledge-like behaviour will succeed. Is there an error bound for the sensors that still allows to escape by a pledge-like behaviour?

¹All segments that are not part of the boundary of some P_i .

The general idea is as follows. We would like to design a class of curves \mathcal{K} of the configuration space. Such a curve will be computed by an agent with sensor errors and imprecise measurements. Any curve from \mathcal{K} represents a possible path for the escape. As mentioned above, the agent is guided by the computed curve and moves close to it. We would like to define sufficient conditions for the curves, such that the escape is guaranteed, if a corresponding path exist. The curve always have precise orientation. Its computation might be erroneous.

As the curves guides the agent, for convenience, we can consider curves in the configuration space by $C(t) = (P(t), \varphi(t))$ where $P(t) = (X(t), Y(t))$ is the location in the work space and $\varphi(t)$ the current turning angle. Note that two full turns will result in a turning angle of 4π instead of zero. Therefore $\varphi(t) \in \mathbf{R}$! A configuration (x, y, φ) is **half-free**, if the curve touches an obstacle in the work space and **free**, if no obstacle in the work space is met. The set of all half-free points is denoted by C_{half} whereas the set of all free points is denoted by C_{free} . Points, where the curve moves from C_{free} to C_{half} are called Hit-Points, h_i . Points, where the curve leaves C_{half} and enters C_{free} are denoted as Leave-Points, ℓ_i . For simplicity, we will also denote the corresponding time parameter by h_i or ℓ_i , respectively.

The Pledge-Algorithm has two movement modi. Either the agent follows the wall and counts turning angles or the agent moves in the free space. Both movements can be erroneous, the agent diverges from the starting direction in the free space and drifts off or the agent cannot count turning angles precisely and will leave the obstacle earlier or later than in the original pledge path.

In principle we have to avoid that the agent moves in infinite loops. The figure shows that a large drift can easily result in a loop. The error of the agent is too large. But also a small deviation in the free space after each leave-point can sum up to a large total drift and an infinite loop. Figure 2.3 shows an example where there are small drifts after each leave that finally results in an infinite loop. This means that the direction in the free space should be globally stable.

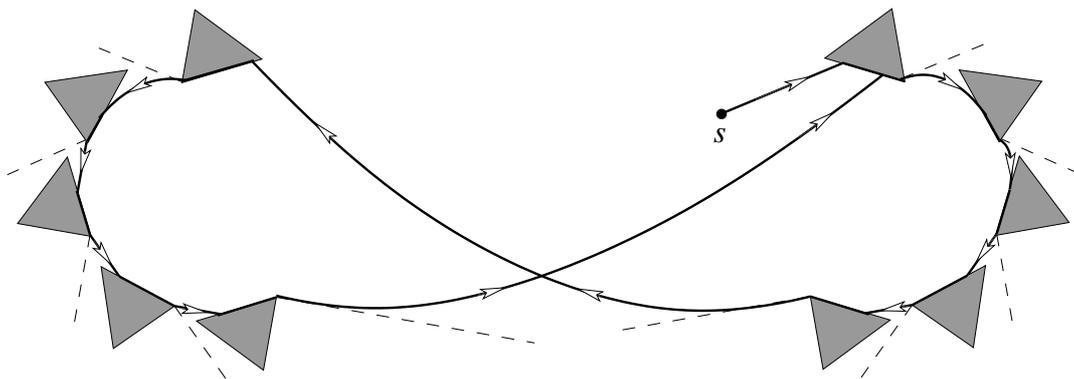
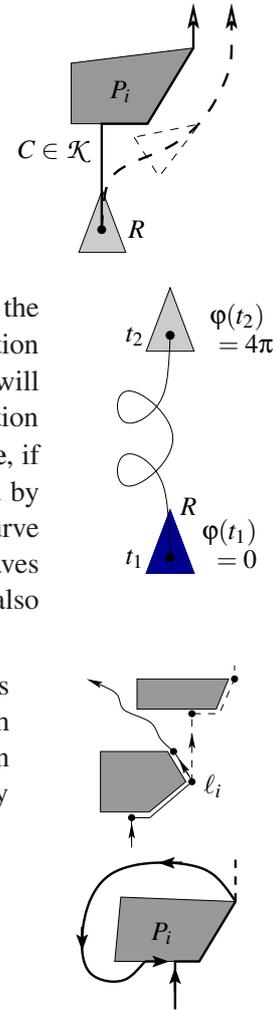


Figure 2.3: Small deviations sum up to a large deviation.

Additionally, for leaving the environment it is highly recommended that the agent at least moves into a certain global direction. One might also think that an erroneous compass will at least allow us to move generally into a half-plane. Therefore we require a C_{free} -condition:

$$\forall t_1, t_2 \in C : P(t_1), P(t_2) \in C_{\text{free}} \Rightarrow |\varphi(t_1) - \varphi(t_2)| < \pi$$

which guarantees that the angular counter in the free space maximally differs from the starting direction $\varphi(s)$ only by a fixed value. If we make use of a compass it seems to be reasonable to think that we can guarantee $\forall t : \varphi(t) \in]-\frac{\pi}{2}, +\frac{\pi}{2}[$ in the free space for starting direction zero.



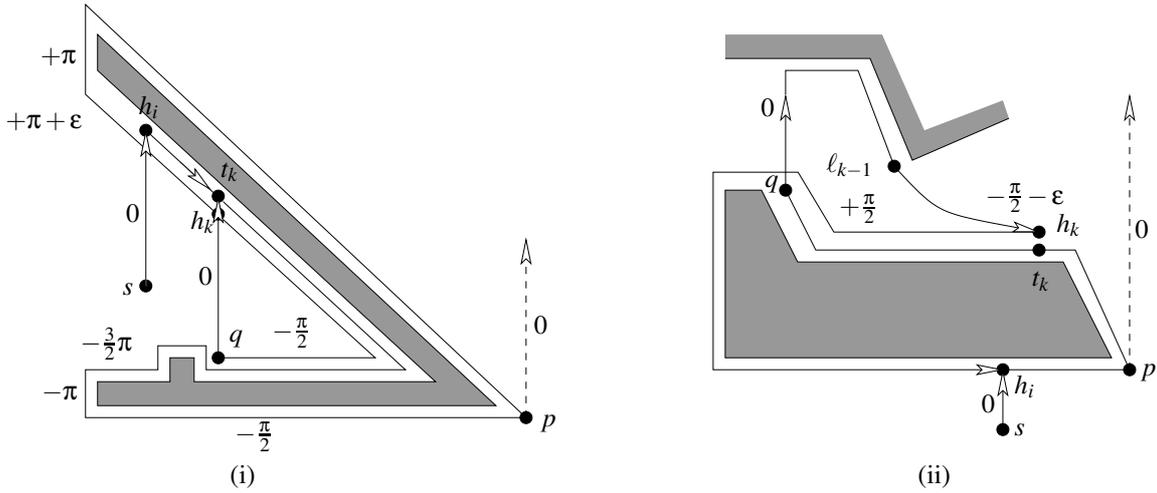


Figure 2.4: A local overturn of the angular counter can result in infinite loops.

Unfortunately, the C_{free} -condition is not sufficient. We have to combine it with the angular counter at the obstacles. Figure 2.4 shows two examples where the agent overturns the counter at the obstacle for a while (because of sensor errors) but obtains overall precise values later for the leaving condition. This has nothing to do with the free space condition. The infinite loop happens at the obstacle. In this case the agent passes the first true leave-point p and leaves the obstacle later at a point q which is also a legal leave-point.

In Figure 2.4(i) by the overturn the curve meets the same obstacle h_k at again. In Figure 2.4(ii) by the overturn the agent first visit another obstacle but then returns via some leave-point ℓ_{k-1} again to the starting obstacle at h_k . In both case $P(h_k)$ is visited twice and during the first visit at t_k the angular counter was overturned. In the left hand side figure the angular counter at t_k is larger than π , in the right hand side figure the counter at t_k is $+\frac{\pi}{2}$. Figure 2.4(ii) shows a second error source. Since the angular counter at h_k is $-\frac{\pi}{2} - \epsilon$ both errors also sum up to an error larger than π . We state that for the hit-point h_k the angular counter of a previous visit was overturned. Together with the error at h_k there is an overall overturn larger than π .

We subsume the requirement in the C_{half} -condition:

$$\forall h_i, t \in C : P(t) = P(h_i) \Rightarrow \varphi(t) - \varphi(h_i) < \pi.$$

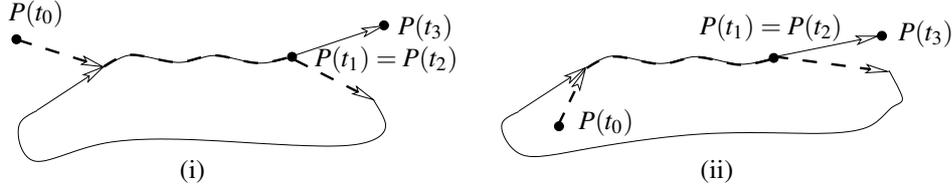
We can also think about the compass with a deviation of $\pi/2$ into both directions. This means that we can overturn the angle counter at the obstacle by less than $\pi/2$. Together with a deviation of less than $-\pi/2$ in the free space we might hit this point again but $\varphi(t) - \varphi(h_i) < \pi$ holds.

Definition 2.5 Let \mathcal{K} be a class of curves in $C_{\text{free}} \cup C_{\text{half}}$ such that any curve from \mathcal{K} fulfils the following conditions:

- (i) Parameterized pledge like curve with turn-angles and position:
 $C(t) = (P(t), \varphi(t))$ with $P(t) = (X(t), Y(t))$
- (ii) At the boundary the curve surrounds obstacles by Left-Hand-Rule.
- (iii) Any leave-point is a vertex of an obstacle.
- (iv) C_{free} -condition holds: $\forall t_1, t_2 \in C : P(t_1), P(t_2) \in C_{\text{free}} \Rightarrow |\varphi(t_1) - \varphi(t_2)| < \pi$
- (v) C_{half} -condition holds: $\forall h_i, t \in C : P(t) = P(h_i) \Rightarrow \varphi(t) - \varphi(h_i) < \pi$

Obviously, the path constructed by the error-free Pledge-Algorithm does belong to \mathcal{K} . For the correctness proof that any curve of \mathcal{K} that is constructed in a pledge-like fashion will escape from the labyrinth. We show some important structural properties of curves in \mathcal{K} .

Lemma 2.6 A curve $C \in \mathcal{K}$ has no self-intersection.

Figure 2.5: The difference between (i) crossing and (ii) touching at t_2 .

Note that a curve from \mathcal{K} might touch itself; see Figure 2.5.

Proof. Assume that C has an intersection, consider the first intersection. There are parameter t_1 and t_2 with $t_1 < t_2$ and $P(t_1) = P(t_2)$ and the first intersection occurs at $P(t_2)$.

This means that between t_1 and t_2 there is a cw or ccw turn. If the intersection lies in the free space, obviously the C_{free} -condition does not hold. Thus we can assume that $P(t_2)$ is in C_{half} holds. Consider the case of a cw loop as depicted in Figure 2.6.

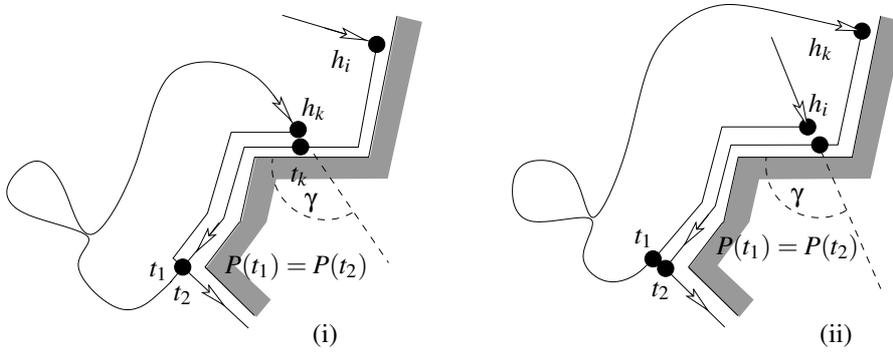


Figure 2.6: Cw loop and two cases.

The curve hits the obstacle at h_i , follows the wall, leaves the obstacle at time t_2 comes back to the obstacles at $h_k > h_i$ and has an intersection at t_2 ; see Figure 2.6(i). If the point $P(h_k)$ was not visited between h_i and t_1 there is only a touching event at t_2 ; see Figure 2.6(ii).

Let $\varphi(h_k^+)$ denote the angular counter after the agent turns into the direction of the corresponding edge. we have $\varphi(h_k^+) = \varphi(h_k) + \gamma$, where γ denotes the turning angle at the edge. We have $-\pi < \gamma < 0$. Additionally, by the full cw turn we conclude $\varphi(h_k^+) = \varphi(t_k) - 2\pi$. Also the C_{half} -holds and we obtain:

$$\begin{aligned} \varphi(t_k) - \varphi(h_k) &< \pi \\ \Leftrightarrow \varphi(h_k^+) + 2\pi - \varphi(h_k) &= \varphi(h_k) + \gamma + 2\pi - \varphi(h_k) < \pi \\ \Leftrightarrow \gamma &< -\pi \quad \zeta \end{aligned}$$

Similar arguments hold for a loop in ccw order, which is an exercise. The first intersection cannot exist. By induction there is no intersection at all. \square

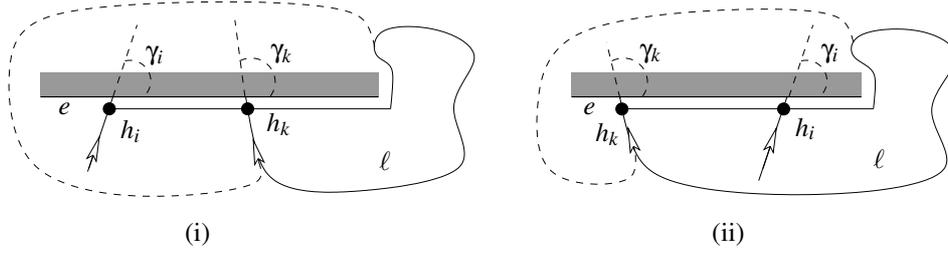
Lemma 2.7 *A curve $C \in \mathcal{K}$ hits any edge of the environments at most once.*

Proof. Assume that a single edge e has two hit-points. After the first hit h_i of edge e the curve can only leave the obstacle at a vertex and then comes back to e at h_k ; see Figure 2.7.

In $P(h_i)$ and $P(h_k)$ the agent turns clockwise in order to follow the edge e which gives turning angles $-\pi < \gamma_i, \gamma_k < 0$. Let $\varphi(h_i^+)$ and $\varphi(h_k^+)$ denote the turning angles as in the proof of Lemma 2.6 (turning angle directly after the hit-point). Let w. l. o. g. $\varphi(h_i^+) = 0$.

In h_i^+ and h_k^+ the curve follow the edge e , and the direction differs only by $2j\pi$ for some $j \in \mathbb{Z}$ This means $\varphi(h_k^+) = 2j\pi, j \in \mathbb{Z}$.

For $j \neq 0$, and with $\varphi(h_i) = -\gamma_i$ and $\varphi(h_k) = \varphi(h_k^+) - \gamma_k$ we conclude $|\varphi(h_k) - \varphi(h_i)| = |2k\pi - \gamma_k + \gamma_i| > \pi$, which contradicts to the C_{free} -condition.

Figure 2.7: A curve from \mathcal{K} hits any edge once.

Therefore we conclude $j = 0$ and also $\varphi(h_k^+) = 0$. But we can argue that there be a full cw or ccw turn from $P(h_i)$ to $P(h_k)$ without intersections; see Figure 2.7. The curve C made a full turn with angular counter change of $\pm 2\pi$. This means that $\varphi(h_k^+) = \pm 2\pi$ should hold. \square

By Lemma 2.7 we can now prove that the condition from Definition 2.5 are sufficient. First, we require a helping lemma. If the curve gets stuck onto a single obstacle, the obstacle should enclose the curve.

Lemma 2.8 *If a curve $C \in \mathcal{K}$ does not leave an obstacle anymore the curve is enclosed by the obstacle.*

Proof. If the curve does not leave an obstacle after the last hit-point, the path along the boundary is repeated infinitely often. The path can be in cw or in ccw order which means a counter change of $+2\pi$ for ccw order or a counter change of -2π in cw order for any round. In the first case at some point the counter gets arbitrarily large for any point on the boundary. So also for the last hit-point and the C_{half} -condition is violated. This means we can only have a cw order loop and by the Left-Hand-Rule the curve has to be enclosed by the obstacle. \square

Theorem 2.9 (*Kamphans, Langetepe, 2003*)

An agent, who follows a path from $C \in \mathcal{K}$ will escape from any labyrinth and from any position, if an escape path exists. [KLO3]

Proof. If there is an escape path the agent and the curve is not enclosed by an obstacle. Therefore the curve $C \in \mathcal{K}$ will leave any obstacle after a while. Since any edge is hit at most once by Lemma 2.7 there will be no hit any more after a while. The C_{free} -condition takes care that the agent steadily moves into a halfplane w.r.t. a given direction. Thus we will escape from the environment. \square

2.1.3 Applications

We would like to consider the impact of Theorem 2.9. Let us first assume that we make use of a compass for counting the total turns around the obstacles and for holding the direction in C_{free} . If the deviation from the starting direction is never larger than $(-\pi/2, \pi/2)$, such a compass will help us leaving the labyrinth.

In this case in C_{free} we can guarantee an angular range $(-\frac{\pi}{2}, +\frac{\pi}{2})$. Along the boundary the absolut error is smaller than $|\frac{\pi}{2}|$, the maximal positive value along the boundary is smaller than $+\frac{\pi}{2}$, the minimal value in C_{free} is larger than $+\frac{\pi}{2}$, this yields the C_{half} -condition.

$$\forall h_i, t \in C : P(t) = P(h_i) \Rightarrow \varphi(t) - \varphi(h_i) < \pi!$$

Corollary 2.10 *The usage of a compass with absolut deviation smaller than $\frac{\pi}{2}$ will help to leave a labyrinth by a pledge like algorithm.*

Next assume that we would like to make use of small deviations of the environment itself. Therefore, we consider obstacles that consists of axis-parallel edges, only. For such orthogonal polygons, we can

simply sum up the reflex vertices (inner angle $> \pi$, $+1$) and the convex vertices (inner angle $< \pi$, -1) as indicated and will leave an obstacle with counter value 0; see Figure 2.8(i). If we guarantee a deviation in the range $(-\frac{\pi}{2}, +\frac{\pi}{2})$ in the free space, we can successfully apply the pledge algorithm. After a hit, we only have to find out whether the edge is horizontal or vertical. For vertical edges we simply slip along the edge and wait for the next hit. Thus we start the movement along the boundary with angular counter $+1$, following a horizontal edge.

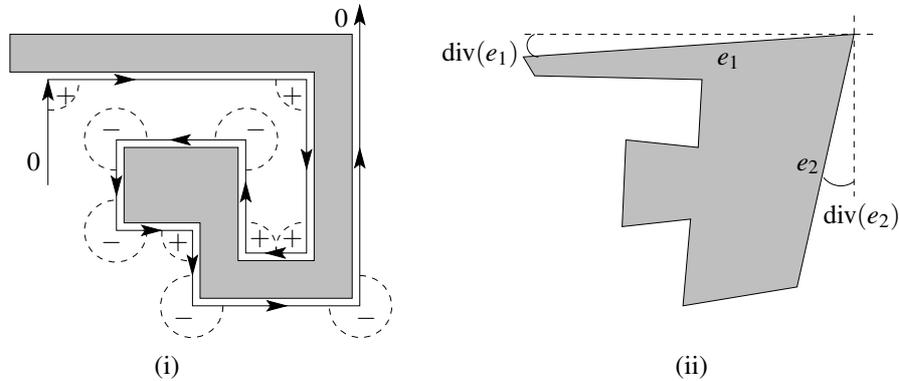


Figure 2.8: (i) Counting the angles in orthogonal polygons, (ii) pseudo-orthogonal polygon with deviation div .

More generally for such a counting argument we have to take care that: Folgende drei Bedingungen sind also einzuhalten:

1. Reflex and convex vertices can be distinguished: Count the rotation correctly.
2. Maximal deviation from the starting direction. Interval of size π .
3. Distinguish: Horizontal/verticale edge.

Now let us assume that the polygons are not exactly axis-parallel but roughly as shown in Figure 2.8(ii).

By, $\text{div}(e)$, for edge Kante $e = (v, w)$ we define the smallest deviation from a vertical or horizontal edge passing through w or v . This deviation should be small in total for all edges. Additionally, we would like to take care that the number of reflex and convex vertices of the polygon fits to an axis-parallel polygon.

Definition 2.11 A polygonal scene is δ -pseudo orthogonal for $\delta > 0$, if for any polygon we have (number of convex vertices) = (Number of reflex vertices) + 4 and $\text{div}(P) := \max_{e \in P} \text{div}(e) \leq \delta$ holds.

For δ -pseudo orthogonal we would like to proceed as indicated above and have to fulfil the above three conditions. Let us additionally assume that the angle counter device on the obstacle has a deviation (error) of no more than ρ . The following corollary shows some legal value for ρ and δ also with the interplay of the free space movements.

Corollary 2.12 Let P be a δ -pseudo orthogonal scene and let us assume that we count the angles with precision ρ such that $\delta + \rho < \frac{\pi}{4}$. If we take care that the deviation from the starting direction in C_{free} is no more than $\frac{\pi}{4} - 2\delta - \rho$, the simple reflex/convex counter pledge like algorithm helps us to leave a labyrinth.

Proof. We have to distinguish between reflex and convex vertices at the boundary, otherwise the $+1$, -1 counting will be erroneous. We consider the worst-case situation for our measurement. Let us assume that at a vertex we measure the outer angle γ as shown in Figure ?? . For $\gamma < \pi$ we assume that we have a reflex vertex and for $\gamma > \pi$ we assume that the vertex is convex.

We would like to guarantee a correct detection for the 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:

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

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 edge 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 .

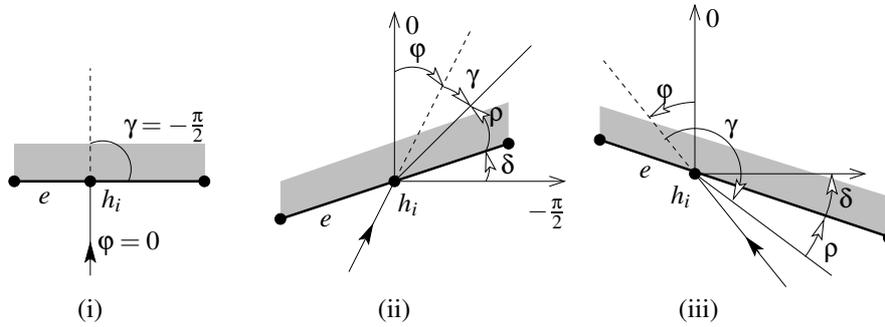


Figure 2.9: Hitting a horizontal edge (i) in the error-free case, (ii) for small absolute γ , (iii) for large absolute γ .

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 detect horizontal edges precisely if $\varphi(h_i) \in]-\frac{\pi}{4} + \delta + \rho, \frac{\pi}{4} - \delta - \rho[$ 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. \square

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 environments². The first

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

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

$\dot{\cup}$	<i>see</i> disjoint union	<i>Duncan</i>	35, 37
1-Layer	14	E	
1-Offset	14	error bound	43
2-Layer	14	F	
2-Offset	14	<i>Fekete</i>	30
<hr/>		G	
lower bound	5	<i>Gabriely</i>	27, 29
A		grid-environment	8
<i>Abelson</i>	43	gridpolygon	8 , 30
accumulator strategy	31	H	
adjacent	8	Hit-Point	50
<i>Albers</i>	30	Hit-Points	44
angular counter	41	I	
approximation	30	<i>Icking</i>	5, 18, 21
<i>Arkin</i>	30	<i>Itai</i>	8
B		J	
Backtrace	19	Java-Applet	18
<i>Betke</i>	30	Java-Applets	41
Bug-Algorithms	49	<i>Jenkin</i>	40
C		K	
cell	8	<i>Kamphans</i>	5, 18, 21, 47
$C_{\text{free-condition}}$	44	<i>Klein</i>	5, 18, 21
$C_{\text{half-condition}}$	45	<i>Kobourov</i>	35, 37
columns	29	<i>Kumar</i>	35, 37
competitive	35, 37	<i>Kursawe</i>	30
configuration space	44	L	
constrained	31	<i>Langetepe</i>	5, 18, 21, 47
Constraint graph-exploration	31	Layer	15
D		layer	27
DFS	8, 11	Leave-Point	50
diagonally adjacent	8 , 27	Leave-Points	44
<i>Dijkstra</i>	19	<i>Lee</i>	19
<i>diSessa</i>	43		
disjoint union	15		
<i>Dudek</i>	40		

