Thijs Vogels

DYNAMIC PATH PLANNING FOR A BASIC ROBOT

Honors Thesis by Thijs Vogels
supervised by Prof. Dr. Henk Meijer
in Spring 2014 at University College Roosevelt

1 | Introduction

Robot path planning is an active area within the field of mobile robotics. The problem of navigating a robot from a starting point S to a target T has proven to be relevant to many applications in real life and it is definitely not trivial.

Current research in the field is mostly focused on the development of intelligent learning robots that navigate through a scene as efficiently as possible by using a variety of sensory information and dynamically collecting data about the environment [9] [6] [15]. In many cases the navigation algorithms found on board of modern robots are heuristic [13]: although their average performance in terms of navigation time or path length tends to be high, there is no guarantee for the robot to reach its target within a bounded path-length.

These modern works are in contrast with the work of early pioneers in the field. Starting with contributions by Lumelsky [2] [7] [8] and Papadimitriou and Yannakakis [1], research in the late 1980's started investigating navigational possibilities for robots with a diverse range of capabilities and requirements. Some of the work indeed studies robots that either learn their environment [15] [22], have complete knowledge about any obstacles in the navigation space [5] [12] or receive information about the environment through visual sensors [9] [10] [19]. On the other hand, there are also studies that focus on more primitive and fundamental robots.

In [7], two navigation algorithms for a robot with only three registers of memory that receives sensory feedback only when it actually hits an obstacle are evaluated. They belong to the category of fundamental problems. Both algorithms are concluded to guarantee reaching the target and the paper presents bounds on the ratio between the path taken and the optimal path under full knowledge of the scene.

This paper can be seen as a continuation of early research in robot motion planning. We study the simplest possible robot that could ever be able to find its way to a target: it is blind and memoryless and only remembers its coordinates and the coordinates of the target. The robot's objective is to travel from a starting point S to a target T in a two-dimensional scene consisting of a finite number of convex impenetrable obstacles. We assume the robot to be a point automaton, such that it can 'squeeze' through touching obstacles. The robot only receives sensory feedback when it hits an obstacle. It can then follow the boundary of the obstacle without knowing about the obstacle's shape.

We study our own algorithm, *BasicAlg*, by which the robot will always try to move in a straight line to T. When the robot hits an obstacle, it will follow the outline of the obstacle

in the direction that initially minimizes the distance to the target until a straight path to T is clear again. It then leaves the outline of the obstacle and repeats its behavior. We assume without loss of generality that if the robot hits an obstacle perpendicularly, the robot moves counterclockwise along the obstacle's boundary. This robotic behavior can be categorized as memoryless and dynamic, since it does not plan its path in advance, but decides on its direction every time it touches an obstacle, solely based on its current position, the position of the target and the gradient of the obstacle it hits.

The paper starts with a literature review of earlier work on this topic and two similar algorithms by Lumelsky and Stepanov [7] in particular. In the following sections, we will evaluate the behavior of our algorithm for cases when the obstacles in the scene are all axis-aligned squares, circles, and similar same-orientation triangles respectively. For squares we provide a tight upper bound on the ratio $\rho = |R(S,T)|/|O(S,T)|$ over all possible scenes, where $|R(S,T)|$ is the length of the path taken by the robot and $|O(S,T)|$ the length of the optimal path from S to T. We find numerical bounds for that ratio for a scene with circles, but these bounds are not tight. For triangles, the ratio is not bounded, but we prove that the robot will always reach the target if it follows our algorithm.

As a student, my role in this project was to (1) conduct a literature survey into previous work on robotic path planning and (2) to extend on the earlier work of Prof. Dr. Henk Meijer and Marijke Hengel by looking into the algorithm for scenes with circles and triangles. I found the proof for guaranteed reachability of the target when the obstacles are similar same-orientation triangles under Henk Meijer's supervision.

2 | Related Theoretical Work

The first classification that can be applied to the broad field of motion planning is that of *off-line* and *on-line* algorithms. *Off-line* algorithms are provided complete information about the scene. The challenge for an off-line algorithm is to find the optimal path for the robot through a scene with a number of known obstacles. These algorithms can be used for robots that are supplied with a map of their surroundings. The paths can be planned in advance, without the robot moving at all. The issues that are dealt with in this branch of motion planning are often related to computational complexity [8] [23] [24] and with approximations for the shape of the scene and the robot [3]. Because most of the algorithms make use of connectivity graphs, all shapes must be approximated by polygons in order for the planning algorithms to run in limited time and space. *On-line* algorithms on the other hand are implemented in behavior of a robot. The algorithms are therefore also termed 'dynamic'. The algorithms run continuously throughout the path of the robot and use the robot's current position and sensory information as input. Of course, off-line algorithms can give more guarantees than on-line algorithms, but most often, complete information is unavailable. In such situations, one must resort to a on-line algorithm. Another advantage of on-line algorithms is that they are mostly not computationally intensive: they often rely on simple choices that are to be made continuously. This paper's direction of choice is into dynamic (i.e. on-line) algorithms. The algorithm that is proposed in this paper is also classified as on-line.

A second subdivision that can be made is in the requirements for the robot's sensors. A substantial part of research is done into robots that have a vision sensor, and therefore have complete information on the part of the scene they look at. These algorithms tend to combine off-line methods for local optimality with a dynamic approach for the global algorithm [10] [3]. Early work on navigation with visual information was carried out by Sutherland [20] and Lumelsky & Skewis [21], and research into navigation with visual information is still an active topic [9]. The alternative to the use of vision sensors are touch sensors. This type of robot only receives feedback the moment it touches an obstacles. Our algorithm falls in the second category, which is therefore more relevant to this paper.

Thirdly, we apply a classification introduced by Kareti *e.a.* [3]. They divide robot navigation research into three classes:

- The goal of research that is classified as *Class A* is to guarantee a certain navigation objective. This objective could be drawing a map of the surroundings, navigating to a goal (point or wall) or anything else that requires navigation. It is not important that parameters such as the distance traveled are minimized, as long as the goal is reached.
- A *Class B* method needs to optimize some parameters. These could be for example the distance traveled, as in [11], or the ratio between the length of the path taken by the algorithm and the optimal path, like in [1] or this paper.
- *Class C* is concerned with computational issues. A *Class C*-paper could for example look into which problems can be solved by a robot with the computational power of a finite state machine. In a way, our work relates to this category in the sense that we study a memoryless robot. We explore the boundaries of what scenes can be successfully navigated through by these robots.

Finally, there is a division between heuristic and non-heuristic algorithms. Non-heuristic robot navigation algorithms can guarantee reaching a target in certain situations, possibly even within a certain bound whereas heuristic algorithms may fail to converge in some cases. Although guaranteed convergence is of course often desirable, it turns out that heuristic algorithms for motion planning often perform better than non-heuristic algorithms [4] . Although heuristic algorithms may perform badly or fail in some situations, they can still perform significantly better *on average* than their non-heuristic variants.

Two Algorithms

This section introduces two algorithms that have been presented by Lumelsky and Stepanov [2] [8] for robots with similar capabilities as our basic robot that can follow *BasicAlg*. The algorithms are dynamic and can be followed by a robot without knowledge of the environment that only receive sensory feedback when hitting an obstacle. In both cases, however, a few registers of memory are required. In that sense, the robots are less basic than the one we will study in this paper. For both of the algorithms, we will give information about the bounds that Lumelsky and Stepanov founds for their performance and in what cases they are non-heuristic.

The first algorithm is called *Bug1*. It uses three registers, say R_1, R_2 and R_3 to store intermediate information. When free, the robot moves towards T. If the target is reached, the procedure stops. When an obstacle is encountered, define a hit point H_j. Now move along the boundary of the obstacle that was hit in a fixed local direction. Assume without loss of generality that this is always clockwise. If the robot reaches the target, stop. Travel around the boundary of the obstacle until back at H_j. Along the way, keep track of three things: (1)

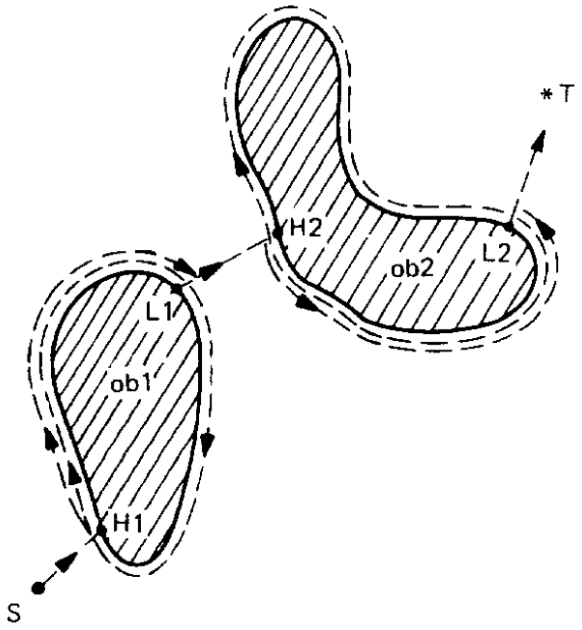

Fig 2.1: Illustration of the *Bug1*-algorithm. By going around the outline of an obstacle between 1 and 1.5 times in order to leave the obstacle at a point closest to the target, this algorithm reaches guaranteed convergence for any scene in which S and T are not enclosed by an obstacle. The algorithm uses three registers of memory.

store L_j, the point that is closest to the target, in register R_1, (2) store the distance traveled along the boundary in R_2, and the distance traveled since L_j in R_3. Using the information in the latter two registers, we can now navigate along the boundary in the most efficient way to leave the obstacle at L_j and move towards T again. The algorithm is illustrated in Fig. 2.1.

Two theorems are used to access the performance of the *Bug1*-algorithm. Firstly, Lumelsky [2] finds that for any algorithm with the given information and robot capabilities and for any $\varepsilon > 0$, there exists a scene for which the length $|R(S,T)|$ of the path generated by the algorithm will obey the relationship $|R(S,T)| \geq d(S,T) + \Sigma\, p_i - \varepsilon$, where $d(S,T)$ is the distance between S and T and $\Sigma\, p_i$ is the sum of perimeters of the obstacles intersecting the line that connects S and T. Secondly, the length of a path generated by *Bug1* never exceeds the limit $|R(S,T)| = d(S,T) + 1.5 \cdot \Sigma\, p_i$. The algorithm is guaranteed to converge for scenes with arbitrary obstacles as long as the target and starting point are not enclosed in an obstacle.

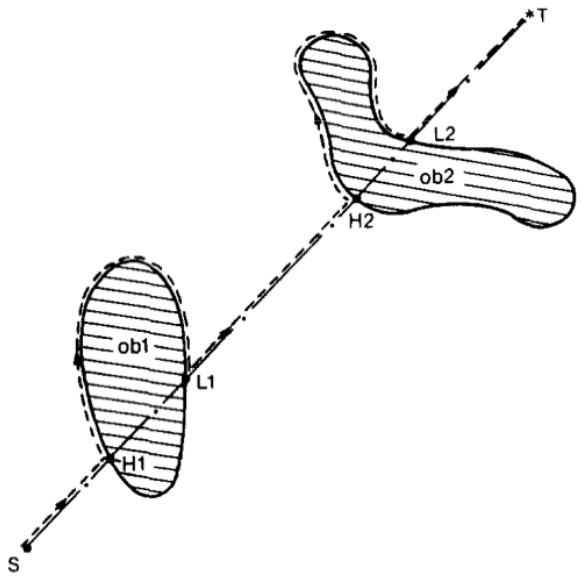

Fig 2.2: Illustration of the *Bug2*-algorithm. This algorithm solves the issue of guaranteed convergence to the target by keeping the ST-line in memory. After following an obstacle, *Bug2* will approach T on that line. This method does not perform well when the scene contains concave obstacles that cross ST often.

Now, let's look at *Bug2*, the second algorithm proposed by Lumeksly and Stepanov. The algorithm goes as follows: Follow the line between S and T. When the target is reached, stop. When an obstacle is encountered, define a hit point H_j and move around the obstacle in a fixed local direction. Assume the local direction without loss of generality to be clockwise. Move along the boundary of the obstacle until either the target is reached or the robot crosses the line segment between S and T. In the latter case, move on towards T. The algorithm is illustrated in Fig. 2.2

This algorithm too needs some memory to keep track of where the line that connects S and T is exactly. Although this algorithm intuitively produces shorter paths than *Bug1*, it can cycle around obstacles multiple times when the obstacles are concave and intersect the line segment ST multiple times. These cycles clearly lead to suboptimal performance. A worst-case bound on the performance of the algorithm is given by $|R(S,T)| = d(S,T) + \sum \frac{n_i p_i}{2}$, where n_i is the number of intersections between ST and the i-th obstacle on ST, and p_i, its perimeter. For convex obstacles, the upper bound for the length of the path simplifies to the worst-case $|R(S,T)| = d(S,T) + \Sigma\, p_i$ and on average $|R(S,T)| = d(S,T) + 0.5 \cdot \Sigma\, p_i$. Together

with the universal earlier lower bound presented in the paragraph on the *Bug1*-algorithm, it follows that this bound is tight.

The two algorithms presented are guaranteed to converge for any scene and can be said to perform reasonably well, with a bound based on the perimeters on the obstacles in the scene. The algorithm proposed in this paper, *BasicAlg*, does not have such bounds in all situations, but has the advantage of assuming no robot memory at all. In a way, it goes back to the essence of the navigation problem even more. Furthermore, in some situations, the basic algorithm could even perform better than *Bug1* and *Bug2*.

3 | Notation and Conventions

The remainder of this paper is quite heavy on specific notation and naming. Most of the notation and writing conventions are announced in this section to aid the reader.

When we refer to points in a scene, or more specifically points on the heuristic path taken by a robot, we use capital letters. The x and y-coordinates of a point A are be indicated by A_x and A_y respectively. The starting point and the target have the same symbols S and T in every scene.

Along the robots path, it hits $n \geq 0$ obstacles. These obstacles are called Ω_i, where i enumerates the obstacles from the first to the last. In theory, an obstacle could have multiple indices if the obstacle is hit multiple times. For each of these obstacles, we define a hit point H_i as the point at which the obstacle first touches Ω_i. Similarly, L_i is the point at which the robot gets loose from Ω_i again.

We refer to the full heuristic path taken by our *BasicAlg*-robot from start to finish by $R(S,T)$ and the path between two specific points A and B by $R(A,B)$. The heuristic paths must be distinguished from an optimal path with complete knowledge of the scene: $O(S,T)$. Path lengths are indicate by absolute value signs, as in $|R(S,T)|$ or $|O(A,B)|$. By default, distances are measured in the L_2-metric. If the L_1 distance is required, this is indicated by a subscript: $|R(S,T)|_{L_1}$.

To assess the performance of the heuristic paths, we define two ratios. Firstly, define

$$\rho = \max \frac{|R(S,T)|}{|O(S,T)|} \tag{3.1}$$

over all possible pairs of S and T under all configurations that are under consideration in a specific section. Similarly, define

$$\lambda = \max \frac{|R(S,T)|}{d(S,T)} \tag{3.2}$$

over all possible pairs of S and T under all configurations that are under consideration in a specific section. Note that always $\lambda \geq \rho$.

4 | Equal Size Squares

This section goes through the earlier work by Prof. Dr. Henk Meijer and Marijke Hengel, that was conducted in the setting of an summer project. They investigated the configurations for the space S where all obstacles are *equal size axis-aligned squares*. This section includes their findings. It is the first of three sections that each try to find bounds on the length of the robot path $|R(S,T)|$ for *BasicAlg* in a specific type of scenes.

We will derive a tight bound of $\rho = \sqrt{10}$ on the length of the robot path over the optimal path, but first, we will show that if there is only one obstacle, then $\rho < 3$ and for any, however small $\varepsilon > 0$ there are are configurations such that the ratio $|R(S,T)|/|O(S,T)| > 3 - \varepsilon$. Secondly, we show that if S and T are restricted to lie on the same horizontal or vertical line and there are arbitrarily many obstacles, we also have $\rho < 3$ and that for any $\varepsilon > 0$ there are configurations such that $|R(S,T)|/|O(S,T)| > 3 - \varepsilon$. Finally, all pieces can be brought together in finding that for any scene with obstacles that are axis-aligned unit squares, we have that $3 - \varepsilon < \rho < 3$ for any ε. The structure of the proof is divided into several lemmas.

Lemma 4.1 If the scene contains exactly one square obstacle, we have $|R(S,T)| < 3d(S,T)$.

Proof Let S and T be two points. If ST does not intersect the interior of the obstacle, we have $|R(S,T)| = d(S,T)$. Now define a coordinate system along the sides of the obstacle. If ST intersects two consecutive sides of the obstacle, we have $|R(S,T)| < |R(S,T)|_{L_1} \leq \sqrt{2} \cdot d(S,T)$ since $\sqrt{2}$ is the maximum ratio between the hypotenuse and the sum of the lengths of a right-angle triangle. So in both cases the lemma holds.

Now assume that ST intersects two opposite sides of the obstacle and that the ratio $|R(S,T)|_{L_1}/d(S,T)$ is maximal. Without loss of generality assume that the obstacle is axis-aligned with corners at $(0,0)$ and $(1,1)$, that ST intersects the vertical sides of the obstacle and that $S_x < T_x$. If $S_y < T_y$ we can increase S_y by some small value δ and decrease T_y by δ so that $|R(S,T)|_{L_1}$ remains unchanged and $d(S,T)$ decreases. In this case, the ratio $|R(S,T)|_{L_1}/d(S,T)$ was clearly not maximal. Similarly, the ratio cannot be maximal when $S_y > T_y$, so $S_y = T_y$. If $S_x < 0$ we can increase S_x by some small value δ so that $|R(S,T)|_{L_1}$ and $d(S,T)$ decrease by δ so the ratio $|R(S,T)|_{L_1}/d(S,T)$ increases. Therefore $S_x = 0$ and similarly we show that $T_x = 1$. It is now not hard to see that $|R(S,T)|_{L_1}/d(S,T) < 3$. ∎

Definition 4.1 We say that a path is x-monotone if the x-coordinates of a point that travels along the path do either not decrease or not increase along the way. We define y-monotone similarly.

Lemma 4.2 Consider a scene with obstacles that are axis-aligned rectangles. Let P be a point on the path $R(S,T)$ with $P_x < T_x$ and $P_y < T_y$. Assume that $R(S,T)$ immediately after P travels in direction (a,b) with $a \geq 0$ and $b \geq 0$. If Q is the first point on $R(S,T)$ after P with $Q_x = T_x$ or $Q_y = T_y$ then the path $R(S,T)$ from P to Q is x- and y-monotone.

Proof After the point P, the path $R(S,T)$ will first travel along the line segment PT. If $R(S,T)$ hits a left vertical boundary of an obstacle, it will continue to travel in direction $(0,1)$. If $R(S,T)$ hits a bottom horizontal boundary of an obstacle, it will continue to travel in direction $(1,0)$. In either case, it will continue to travel in direction $(0,1)$ or $(1,0)$ until it either hits the x- or the y-axis, or reaches another point A from which it travels along the line segment AT. In that case we can repeat the argument. We conclude that both the x- and y-coordinates do not decrease. ∎

Lemma 4.3 Consider obstacles that are equal size unit squares. Suppose S and T lie on the x-axis. In this case any point P on $R(S,T)$ satisfies $-1 < P_y < 1$.

Proof Without loss of generality, assume that S and T lie on the $x-$axis with $S_x < T_x$. Consider P_y to be a function whose parameter is a point P that travels along $R(S,T)$ from S to T and whose value is the y-coordinate of P. Observe that the absolute value of P_y can only increase if P lies on the left vertical side of an obstacle (and not on a corner of the obstacle). The point P travels upward if the first point of $R(S,T)$ on this side has a y-coordinate ≤ 0, and downwards if it positive. In both cases it follows that $-1 < P_y < 1$ for all points P. ∎

The above lemma implies that if $S_x = T_x$ and $S_y < T_y$ there is no point P on $R(S,T)$ with $P_y > T_y$: let Q be the first point of $R(S,T)$ with $Q_y = T_y$. Since $-1 < Q_x < 1$ there can be no obstacle between Q and T, so after Q, the path $R(S,T)$ stays on the line $y = T_y$.

Lemma 4.4 If S and T lie on the same horizontal or vertical line, and the obstacles are axis aligned unit squares, we have $|R(S,T)| < 3d(S,T)$.

Proof Let n be the number of obstacles. We prove the lemma by induction on n. If $n = 1$ the lemma holds by Lemma 4.1. Assume that the lemma holds if there are n obstacles for some

$n \geq 1$. Now assume that there are $n+1$ obstacles. Assume without loss of generality that S and T are both on the x-axis with $S_x < T_x$. The path $R(S,T)$ then first travels to the right on the x-axis until it hits the first obstacle at a point H_1. The path then runs upwards along the side of the obstacle. So the path continues until it hits the right top corner L_1 of the obstacle. From L_1 the path travels in a direction (a,b) with $a > 0$ and $b \leq 0$.

By Lemma 4.2 we derive that the path after H_1 and L_1 has monotone x- and y- coordinates until it reaches the x-axis at W or the y-axis at point Z. In the latter case we derive from Lemma 4.4 that $-1 < Z_y > 1$ so $R(S,T)$ from Z to T follows the y-axis and the statement of the lemma holds. If the path reaches W, we know that from the inductive assumption that

$$|R(W,T)| < 3d(W,T). \tag{4.1}$$

Moreover we have

$$|R(S,W) \leq |SH_1| + |H_1L_1| + d_{L_1}(L_1,W) \leq 3d(S,W), \tag{4.2}$$

so

$$|R(S,T)| = |R(S,W)| + |R(W,T)| < 3d(S,W) + 3d(W,T) = 3d(S,T). \tag{4.3}$$

So the lemma holds. ∎

Lemma 4.5 If S and T lie on the same horizontal or vertical line, and the obstacles are equal size axis aligned squares, then, for any $\varepsilon > 0$ there is a configuration for which

$$\frac{|R(S,T)|}{O(S,T)} > 3 - \varepsilon. \tag{4.4}$$

Proof This is a proof by construction. Consider a scene with one unit-size square obstacle positioned at $(0,0),(0,1),(1,1),(1,0)$. Let $S = (0, a\varepsilon)$ and $T = (1, a\varepsilon)$. The optimal path length $|O(S,T)|$ now is $1 + 2a\varepsilon$ and the robot path length $|R(S,T)| = 3 - 2a\varepsilon$. For any positive $a < \frac{1}{8-2\varepsilon}$, the ratio $\frac{|R(S,T)|}{O(S,T)} > 3 - \varepsilon$. ∎

Lemma 4.6 If S and T lie on the same horizontal or vertical line, and the obstacles are equal size axis aligned squares, we have $3 - \varepsilon < \rho < 3$ for any $\varepsilon > 0$.

Proof This follows from Lemma 4.5 and Lemma 4.4. ∎

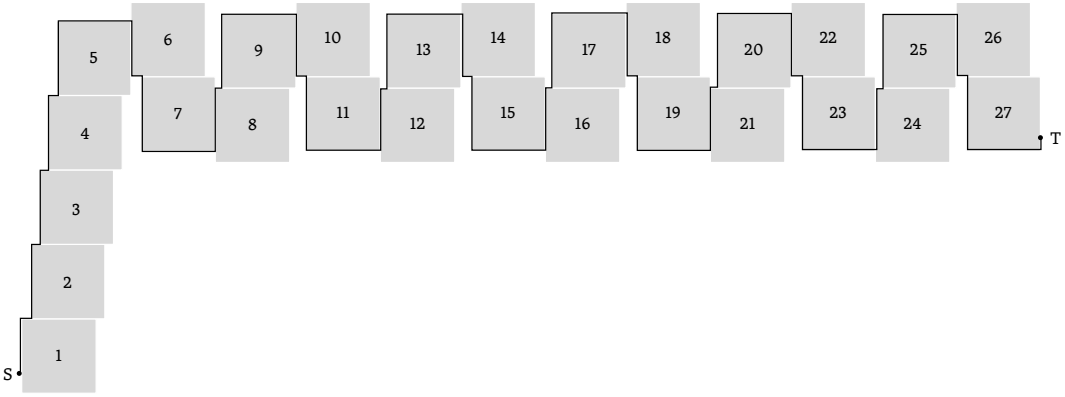

Fig 4.1: Illustration of the constructive proof in this chapter. By placing unit squares in the way illustrated in this figure, a lower bound on the ratio ρ can be set to $\sqrt{10}$.

Lemma 4.7 If the obstacles are equal size axis aligned squares, we have

$$\lambda = \max \frac{|R(S,T)|}{d(S,T)} < \sqrt{10}. \tag{4.5}$$

Proof Without loss of generality, assume that $S_x < 0$, $S_y < 0$ and $T = (0,0)$. From Lemma 4.2 we derive that $(R(S,T)$ is initially x- and y-monotone until it intersects that x- or y-axis for the first time, at point P say. Without loss of generality assume that P lies on the x-axis. Let $Q = (S_x, 0)$. We have

$$\begin{aligned} |R(S,T)| &\leq d_{L_1}(S,P) + |R(P,T)| \\ &< d(S,Q) + d(Q,P) + 3d(P,T) \leq d(S,Q) + 3d(Q,T). \end{aligned} \tag{4.6}$$

Consider the function $f(a,b) = (a+3b)/\sqrt{a^2+b^2}$. By computing the gradient of f and setting it to $(0,0)$, we find that the maximum value of f occurs at $a = 1$ and $b = 3$ and is $\sqrt{10}$. It turns out that $f(a,b) \leq \sqrt{10}$ for all a and b. Let $a = d(S,Q)$ and $b = d(Q,T)$.

$$\frac{|R(S,T)|}{d(S,T)} < \frac{a+3b}{\sqrt{a^2+b^2}} \leq \sqrt{10}. \tag{4.7}$$

This proves the lemma. ∎

Lemma 4.8 If the obstacles are equal size axis aligned squares, then there is a configuration for which

$$\rho = \max \frac{|R(S,T)|}{|O(S,T)|} > \sqrt{10} - \varepsilon \tag{4.8}$$

for any $\varepsilon > 0$.

Proof Let k be a large even integer and let $\delta \leq 1/k$. For an illustration of the construction in this proof, see Fig. 4.1, where $k = 4$. In the figure we used $\delta = 1/8$. Place S at $(0,0)$ and T at $(3k(1+\delta) + (k-1)\delta, k)$. Place $7k-1$ rectangles as follows. Place square 1 with its left-bottom corner at $(0, -\delta)$. For $1 < i \leq k+1$, place square i on top of square $i-1$, shifted δ to the right. Place square $k+2$ to the right of square $k+1$, and shift it up by 2δ. Place square $k+3$ below square $k+2$, and shift it right by δ. Place square $k+4$ to the right of square $k+3$, and shift it down by 2δ. Place square $k+5$ above square $k+4$, and shift it right by δ. As shown in the figure, we keep adding squares in groups of two, by repeating the last 4 placements. Notice that the heuristic path zigzags through the odd numbered squares that have numbers $> k$.

We have $|R(S,T)| \approx k + 3 \cdot 3k = 10k$ and $|O(S,T)| \approx \sqrt{k^2 + (3k)^2}$ so

$$|R(S,T)| \approx \sqrt{10} \cdot |O(S,T)|. \tag{4.9}$$

We can show that we can show k such that the fraction is arbitrarily close to $\sqrt{10}$. We have

$$|R(S,T)| > k + 3k(3 - 2\delta) = 10k - 6k\delta \geq 10k - 6. \tag{4.10}$$

$$|O(S,T)| < 2 + \sqrt{(3k(1+\delta))^2 + k^2} = 2 + k\sqrt{10 + 18\delta + 9\delta^2}. \tag{4.11}$$

Since $\delta \leq 1/k$ we have

$$\lim_{k\to\infty} \frac{10k - 6}{2 + k\sqrt{10 + 18\delta + 9\delta^2}} = \sqrt{10}. \tag{4.12}$$

So the lemma holds. ∎

Theorem 4.1 If the obstacles are equal size axis aligned squares, we have $\sqrt{10} - \varepsilon < \rho < \sqrt{10}$ for any $\varepsilon > 0$.

Proof This follows from Lemma 4.7 and Lemma 4.8 using the fact that $\rho < \lambda$. ∎

Comparing Performance

Comparing the results in this section with the performance of Lumelsky and Stepanov's *Bug1* and *Bug2* algorithms [8], we look at a limiting case. Considering $d(S,T) = |O(S,T)| \to \infty$ and setting the perimeters p_i of the all objects to 4, we find that for *Bug1* $|R(S,T)| \leq d(S,T) + 6d(S,T)$, so $\rho \leq 7$. The performance of this algorithm is in this specific situation significantly lower than that of *BasicAlg*, even though the robot for *Bug1* has slightly higher requirements.

Similarly, we can look at the performance of *Bug2* in this particular set of scenes. We find that for *Bug2*, $\rho \leq 5$. Using Lumelsky and Stepenov's result that $|R(S,T)| \geq d(S,T) + \Sigma\, p_i - \varepsilon$, we can see that at least for *Bug2*, this bound on ρ is tight.

5 | Circles

This section investigates scenes in which all obstacles are circles. Although we could not find exact bounds on the performance of *BasicAlg* in this situation, we have numerical upper- and lower bounds on ρ that were derived by Dr. Meijer earlier. These bounds are not yet tight. Again, we prove the bounds in the structure of a series of lemmas. The structure will be similar to that of the previous section. The first lemma will show that if there is only one obstacle, then $1.086 < \rho < \pi/2$. We also show that is there are an arbitrary number of obstacles then $1.403 < \rho < 1.666$. Lastly we show how with the help of *Mathematica* we can tighten the above bounds.

Lemma 5.1 If the scene contains one circular obstacle, then

$$\lambda = \max \frac{|R(S,T)|}{d(S,T)} \leq \tfrac{1}{2}\pi \tag{5.1}$$

Proof Let S and T be two points. Assume that ST intersects the disk and that the ratio $|R(S,T)|/d(S,T)$ is maximal. If S does not lie on the disk, we can move it a bit closer to T, so both $d(S,T)$ and $|R(S,T)|$ decreases by some infinitesimal value δ. Since $|R(S,T)|/d(S,T) > 1$, this operation increases $|R(S,T)|/d(S,T)$. This shows that the ratio $|R(S,T)|/d(S,T)$ was not maximal.

If T does not lie on the disk, we can move it a bit closer to S, so $d(S,T)$ decreases by some small value δ and $|R(S,T)|$ decreases with less than δ. This operation also increases $|R(S,T)|/d(S,T)$. This shows that the ratio $|R(S,T)|/d(S,T)$ was not maximal. We conclude that both S and T lie on the disk. Now it is easy to see that ST is a diagonal of the disk and $|R(S,T)| = \frac{1}{2}\pi \cdot d(S,T)$. ∎

Lemma 5.2 If there is one circular obstacle

$$\rho = \max \frac{|R(ST)|}{|O(S,T)|} > 1.086 \tag{5.2}$$

Proof Let $S = (2.562, 0)$ with $x > 1$, $T = (-1, 0)$ and let the obstacle be a unit disk centered at the origin. We have $|R(S,T)| = 1.562 + \pi$. Moreover,

$$|O(S,T)| = \sqrt{2.562^2 - 1} + \pi - \arccos(1/2.562). \tag{5.3}$$

This gives

$$\frac{|R(S,T)|}{|O(S,T)|} \approx 1.08614 > 1.086, \tag{5.4}$$

which proves the lemma. ∎

In a configuration with one circular obstacle and a maximal value of $|R(S,T)|/|O(S,T)|$ it is not hard to see that T lies on the obstacle. We used both Mathematica as well as program written in Java to solve this problem, and both implementations computed that in the optimal configuration, we have $x \approx 2.562$, $y = 0$ and $\rho = |R(S,T)|/|O(S,T)| \approx 1.08614$, so this confirms that the bound in the previous lemma is almost tight.

We first prove a lemma we need when computing an upper bound on ρ for an arbitrary number of disks. If A and B lie on a disk then we use the notation $arc(A,B)$ to denote the length of the shortest arc on the disk from A to B.

Lemma 5.3 If S lies on a disk, ST intersects the disk and P is the last point of $R(S,T)$ on the disk, then

$$\frac{arc(S,P)}{d(S,T) - d(P,T)} \leq 1.666 \tag{5.5}$$

Proof Suppose that we have a configuration in which $ratio = arc(S,P)/(d(S,T) - d(S,T))$ is maximal. If ST does not contain the diameter, we move move T away from P along the line through P and T. This increases $d(S,T)$ by less than $d(S,T)$ increases, while $arc(S,P)$ remains equal, so the ratio increases. Therefore ST does contain the diameter of the disk. By placing S at $(-1,0)$, the disk centered at the origin and T at $(x,0)$, we can compute that

$$ratio = \frac{arc(S,P)}{d(S,T) - d(P,T)} = \frac{\pi}{x + 1 - \sqrt{x^2 - 1}}. \tag{5.6}$$

This function is maximal at $x \approx 1.341$ where $ratio \approx 1.6656$. ∎

Lemma 5.4 If there are an arbitrary number of circular obstacles, then

$$\lambda = \frac{|R(ST)|}{d(S,T)} \leq 1.666. \tag{5.7}$$

Proof We prove this lemma by induction on n, the number of obstacles. The lemma holds for $n = 1$ by Lemma 5.1. Assume the lemma holds for $n \geq 1$ obstacles. We now assume to have $n + 1$ obstacles and we have a configuration with the maximal value of $|R(ST)|/d(S,T)$.

As before we can argue that S lies on an obstacle. Assume that S lies on the unit disk centered at the origin. Let P be the last point of $R(S,T)$ on this disk. Then by induction we have $|R(P,T)| \leq 1.666d(P,T)$. From Lemma 5.3 we have that the length of $arc(SP) < 1.666(d(S,T) - d(P,T))$. So

$$\begin{aligned} |R(S,T)| &= arc(SP) + |R(P,T)| \\ &\leq 1.666(d(S,T) - d(P,T)) + 1.666d(P,T) = 1.666d(S,T), \end{aligned} \tag{5.8}$$

which proves the lemma. ∎

Lemma 5.5 There is a configuration for a scene with a number of disks as obstacles for which

$$\frac{|R(ST)|}{|O(S,T)|} > 1.403 \tag{5.9}$$

Proof This is a proof by construction. The construction was carried out in a *Mathematica* program. Fig. 5.1 illustrates the construction and the code is available in the Appendix. We start by placing T at $(t,0)$. Next, we iteratively add circles to the scene. The first circle is a unit circle with its center in $(0,1)$. The other circles are placed such that they 'rest' on the x-axis, they touch the previous circle and they are hit perpendicularly by the robot path. This way, in the limit $t \to \infty$, the optimal path $O(S,T) \to t$ and the robot path goes around most circles. For $t = 2000$, we can put in 19 circles, and the ratio $|R(S,T)|/|O(S,T)| = 1.4037$. ∎

We conclude that $1.403 < \rho < 1.6666$. This bound is not tight. We expect that there should be configurations possible that show a higher ratio than 1.403, but also 1.6666 seems a very generous upper bound.

Comparing Performance

For configurations with circles, our *BasicAlg* again generates a significantly lower upper bound on ρ than *Bug1*. This is illustrated by the scene in which S and T lie on opposite sides a disk, at an infinitesimal distance $\delta > 0$ from its boundary such that ST goes through the middle of the disk. The ratio $|R(S,T)|/|O(S,T)|$ for this situation gets as high as 3. For this particular configuration, *Bug2* performs the same as our algorithm.

An investigation of several other configurations with circles shows that mostly, the performance of *Bug1* relatively weak, while *Bug2* performs very well and the performance of *BasicAlg* performs well in most situations.

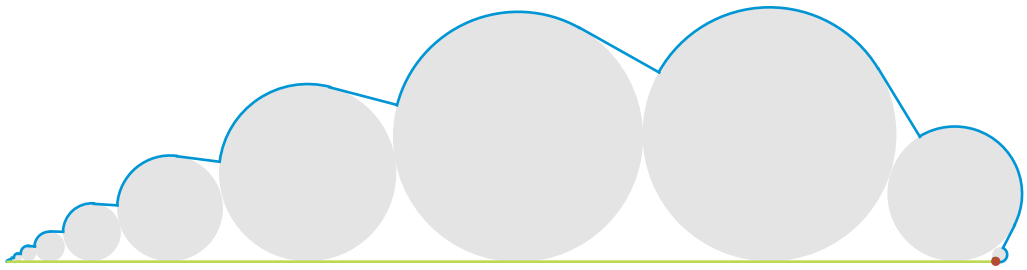

Fig 5.1: Construction of a scene with arbitrary size circluar obstacles in which $|R(S,T)|/|O(S,T)| > 1.403$ following the description in the proof of Lemma 5.5. The target T (in red) is located at $(2000, 0)$, the scene contains 19 circles (although some are pretty small), and the ratio of interest is 1.4037. The robot path is drawn in blue and the optimal path in green (#bada55).

6 | Similar Same-Orientation Sharp Triangles

In the line of the previous sections, this section investigates scenes in which all obstacles are similar same-orientation sharp triangles. Although the ratio ρ under *BasicAlg* is unbounded in this case, we prove that convergence of *BasicAlg* is guaranteed.

Assume all obstacles in S to be similar triangles of the same orientation. We use a coordinate system with T in the origin. The sides of triangle number k are called a_k, b_k, c_k such that all sides a_i have the same direction α, all sides b_i have direction β and all sides c_i have direction γ. Now consider the direction α. We define a half-plane Π_α that is delimited by a line in the direction α through T. It indicates the region in which the robot could hit a triangle at a side in direction α. Define the half-planes Π_β and Π_γ accordingly. The half-planes are well defined because the robot always moves towards T.

Next, define 'axes'. For the direction α, define A_α as a ray from the origin into Π_α that is perpendicular to α. By definition, if a the robot hits an obstacle at the α-side, it will always be in Π_α and decide to move towards A_α. Define A_β and A_γ similarly. Those axes have the same properties.
We aim to prove that if the robot crosses an axis A_x at a Euclidean distance d from T, it will never cross the same axis at a distance $\geq d$ later in time. In order to prove this, we start by introducing some concepts that play a role in the argument.

Definition 6.1 The orthocenter of a triangle is defined as the intersection of its three altitudes (lines through a vertex, perpendicular to the opposite side.)

Definition 6.2 Define a custom measurement M for a point's distance to T. Let Γ_1 be a triangle that is similar to the obstacles but rotated by 180° compared to the obstacles. Let Γ_1 have its orthocenter in T and a circumference of 1. The distance $M(P)$ of a point P to T is defined as the factor $f \geq 0$ with which Γ_1 should be scaled with respect to T such that $p \in \Gamma_1$.

In the above definition, we use the fact that the vertices of Γ_1 are by Definition 6.1 on the axes A_α, A_β and A_γ. An interesting property of the distance measure M is that it is not commutative. This is not problematic, since we only measure the distance from points to a fixed point T.

The heuristic path $R(S, T)$ crosses the three axes 0 or more times. The next definition defines these crossings.

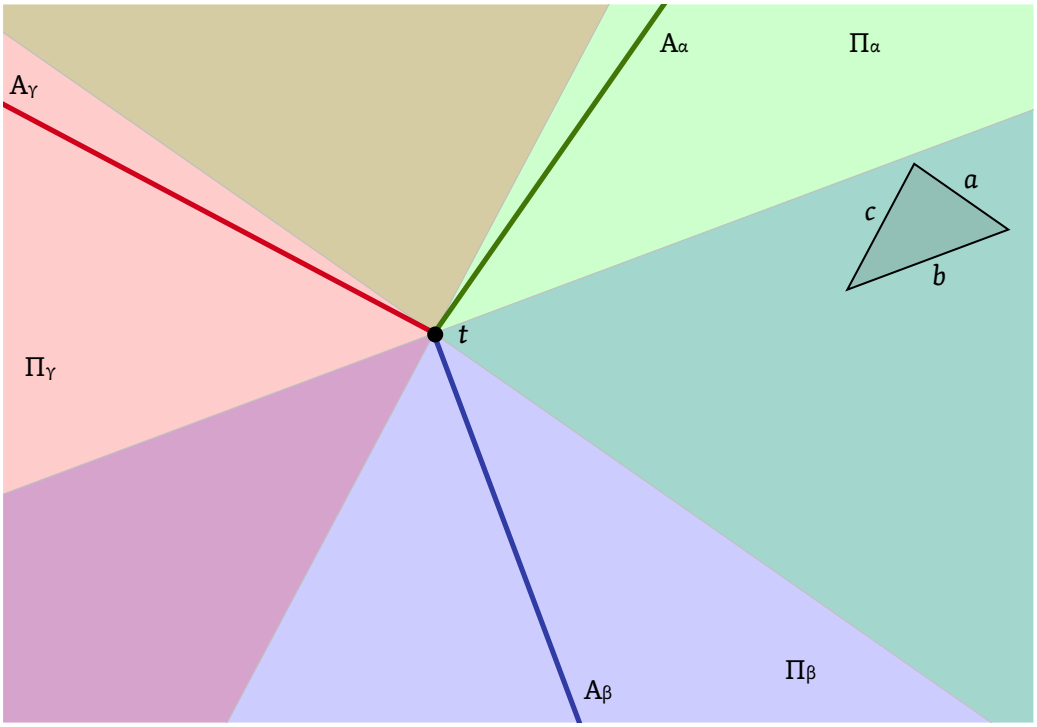

Fig 6.1: Illustration of defined concepts. The half-planes $\Pi_\alpha, \Pi_\beta, \Pi_\gamma$ and axes $A_\alpha, A_\beta, A_\gamma$ are shown for a scene with triangles of orientations a, b, c.

Definition 6.3 Let $\mathbf{C} = \{C_1, C_2, \ldots\}$ be a vector containing those crossing points and let $A(C_i)$ be the corresponding axes and $\Pi(C_i)$ the corresponding half-plane.

Definition 6.4 Consider one of the three half-planes. Without loss of generality, assume this to be Π_α. For a point $P \in \Pi_\alpha$, define $D_\alpha(P)$ as the M-distance between T and the orthogonal projection of P on A_α. $D_\alpha(P)$ is not defined for points P that are not in Π_α. In the course of this proof, we will speak loosely about 'the D_α' of the robot in time. Definitions for D_β and D_γ follow accordingly.

Lemma 6.1 For any $x \neq y$, the intersection $A_x \cap H_y = \{T\}$.

Proof This follows directly from the fact that the obstacles are sharp triangles. For a direction $x \in \{\alpha, \beta, \gamma\}$, the axis A_x is perpendicular to the direction x, whereas the angles between x and the other two directions are smaller than 90°. ∎

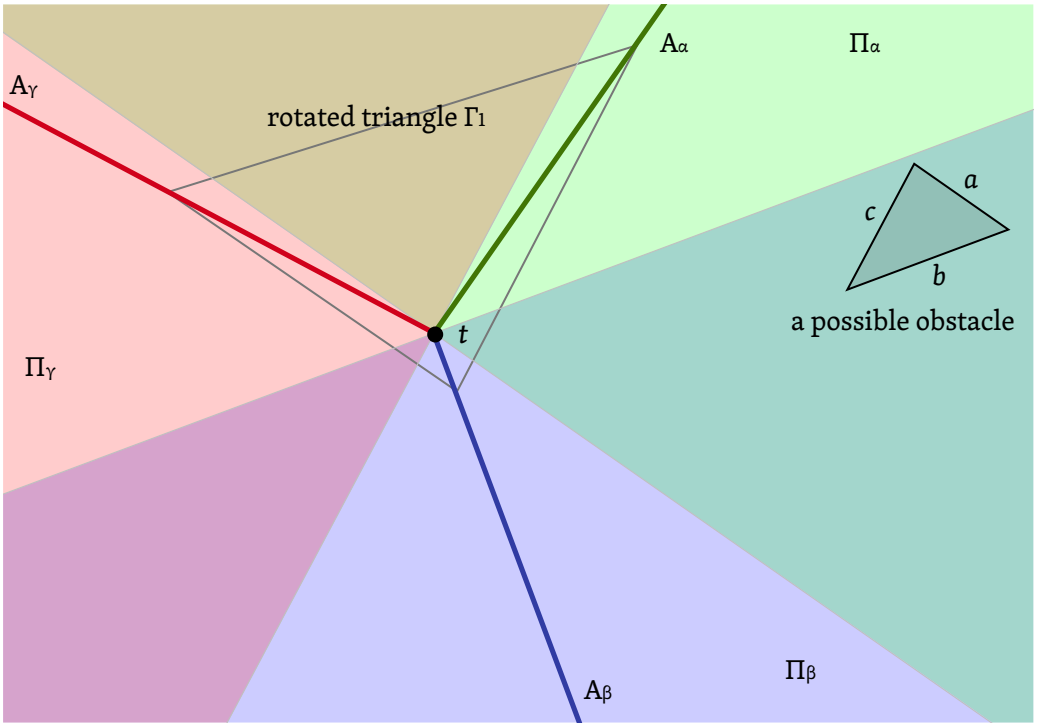

Fig 6.2: M-distance. The gray triangle indicates a set of points of equal M-distance.

Lemma 6.2 For any direction $x \in \{\alpha, \beta, \gamma\}$, looking at the D_x over time in the heuristic path of the robot, D_x can only increase while the robot follows the side of an obstacle Ω_i and it has started following that side at the corresponding hit-point H_i that is not in Π_x.

Proof Suppose the robot starts following the side of a triangle. At the moment it first hits the triangle at H_i, according to Lemma 6.1, it is in one half-plane. Assume this without loss of generality to be Π_α. Suppose that D_α increases while following the side. If the side has the α-direction, D_α will stay constant. If the side is not in the α-direction, D_α decreases, since the robot always aims towards T. This contradicts the assumptions and proves the lemma. ∎

Lemma 6.3 An obstacle Ω can have a non-empty intersection with at most 2 of the half-planes $\Pi_\alpha, \Pi_\beta, \Pi_\gamma$.

Proof This follows from geometric observations. If the obstacle would have a non-empty intersection with all thee half-planes, it would enclose the target T, which is not allowed. ∎

Lemma 6.4 Let Γ be a triangle with sides a, b, c that intersects with two axes. Without loss of generality, let the axes be A_α and A_β. The intersections of side c (with direction γ) with te axes A_α, A_β have equal M-distance.

Proof This follows from the observation that along a triangle's side in direction γ that is not in Π_γ, the M-measure is constant by definition. ∎

Lemma 6.5 Consider a point C_i in the robot path at which the robot crosses an axis. If $M(C_i) = d$ for some i and some distance d, then either $M(C_{i+1}) < d$ (the next axis-crossing is closer by) or $M(C_{i+2}) < d$ (the crossing after that is).

Proof Consider the side s of the obstacle Ω_i that is followed immediately after the robot crosses an axis at C_i. We distinguish two scenarios: (1) at the moment the robot stops following s, the robot is only in one half-plane. Without loss of generality, let this be $\Pi_\alpha = \Pi(C_i)$. In the second scenario (2), after following side s, the robot is both in Π_α and another half-plane. Without loss of generality, let this second half-plane be Π_β.

1. If the robot ends up only in the half-plane Π_α after following the side, it is not free in Π_α and can only hit sides in the α-direction. It will move towards the axis A_α. Since, according to Lemma 6.2, D_α will decrease, the path will cross A_α again, closer to T than before. So $M(C_{i+1}) < d$.
2. If the robot ends up in $\Pi_\alpha \cap \Pi_\beta$ after following s, it will follow another side of Ω_i before leaving its boundary. Assume without loss of genenerality that this side is in the β-direction and call the side ν. The moment after directly following ν, the robot can either be in (1) $\Pi_\alpha \cap \Pi_\beta$ or (2) in Π_β only:

    1. If, when it leaves ν, the robot is still in $\Pi_\alpha \cap \Pi_\beta$, it did not cross A_β (using Lemma 6.1). Note that, if D_β would be $\geq d$, the obstacle Ω_i would intersect with both A_α and A_β. This is in contradiction with the observation that the robot did not cross A_β. We conclude that at this moment, $D_\alpha < d$ and $D_\beta < d$. Since those measures decrease monotonely while the robot is in Π_α and Π_β and the robot now moves towards on of the axes A_α and A_β, it will intersect one of them at $M(C_{i+1}) < d$.
    2. If the robot is now only in Π_β, it will inevitably move towards A_β. Following the same argument as directly above, if the robot did not cross the axis A_β, $D_\beta < d$ and $M(C_{i+1}) < d$. Now we consider the case that the robot did cross A_β. Due to

Lemma 6.3, the robot is now only in half-plane Π_β and will move towards A_β. Although its current D_β could be $\geq d$, it will strictly move towards A_β. Since by Lemma 6.4, the part of the axis with $D_b \geq d$ is covered by Ω_i, the robot is forced to intersect the axis at a distance $M(C_{i+2}) < d$.

This completes the proof for Lemma 6.5. ∎

Theorem 6.1 If the robot crosses an axis A_x at a Euclidean distance d from T, it will never cross the same axis at a distance $\geq d$ later in time.

Proof If for some i, $A(C_i) = A(C_{i+1}) = A_x$, then $M(C_i) > M(C_{i+1})$. This follows from the fact that for any direction $x \in \{\alpha, \beta, \gamma\}$ D_x decreases while the robot is in Π_x, and that the robot must cross another axis if it would leave Π_x.

Furthermore, from Lemma 6.5, we have that if the robot crosses two different axes after each other, it will eventually cross that axis at a smaller M-distance. Suppose the robot comes back to the axis A_x at the k'th crossing C_k after having crossed a series of other axes. Let Ω_k be the obstacle along which the robot crosses at C_k and let H_k be the point at which the robot starts following Ω_k. Assume now that (contrary to the theorem) $M(C_k) \geq d$. Repeatedly using Lemma 6.4 we know that $M(H_k) < d$, therefore, from Lemma 6.4, the obstacle Ω_k will cover the axis A_x all the way between C_k and the point on A_x at M-distance d. This is not possible, since the obstacle Ω_i crosses the axis in this region too and obstacles are not allowed to intersect.

We conclude that if the robot crosses an axis A_x for any $x \in \{\alpha, \beta, \gamma\}$ at some Euclidean distance d from T, it will never cross the same axis at a distance $\geq d$ later in time. ∎

Comparing Performance

In this section, we have proved that a robot following our algorithm *BasicAlg* will always get closer to its target. It is easy to see that, if the number of obstacles in the scene is finite and the obstacles have finite size, this implies guaranteed convergence of our algorithm.

For triangles, there clearly is no bound on the ratios λ or ρ. This is illustrated by the situation where S and T are very close together, with the corner of a huge triangle Γ in between them. It is easy to place the triangle such that the robot is forced to walk along the obstacle. The same is true for *Bug1* and *Bug2*, though. Like *Bug1* and *Bug2*, our algorithm is bounded by the sum of perimeters of the obstacles and $d(S, T)$, since it will follow an obstacle's boundary at most once.

7 | Conclusion

In the light of the early research in robot motion planning, we have gone back to the essence motion planning. *BasicAlg*, the algorithm presented in this paper can be followed by a robot with absolute minimal requirements. It only needs to know its location, the location of the target and it should now when it touches an obstacle.

The performance of this algorithm has been evaluated for three types of scenes: with equal size squares, circles and with similar same-orientation triangles. For the first two categories, we have been able to find a bound on the ratio of the length of the robot path over the length of the optimal path. For each of the situations, we can guarantee convergence to the target. When comparing the performance of our algorithm to that of the *Bug1*- and *Bug2*-algorithms presented by Lumelsky and Stepanov [2], we do not see clear differences. The main advantage of their method is that it can guarantees convergence even for concave obstacles, whereas our algorithm can be run on robots with less requirements.

During writing of this paper, we also obtained a tight bound of $\sqrt{82}$ for the ratio of the robot path over the optimal length in the case of arbitrary size squares. Due to time constraints, the proof is ommitted.

We conjecture that *BasicAlg* is guaranteed to converge for any scene with obstacles that are convex, similar and have the same orientation. A first step towards proving this, would be to extend the proof for sharp triangles into arbitrary triangles, after which it should be generalizable to convex n-gons.

Kareti *e.a.* have put the essence of this work nicely: “These formulations, although abstract and simplified compared to real-life scenarios, provide the basis for practical systems by highlighting the underlying critical issues.” [3]

References

[1] Papadimitriou, Christos H., and Mihalis Yannakakis. "Shortest paths without a map." *Theoretical Computer Science* 84, no. 1 (1991): 127–150.

[2] Lumelsky, Vladimir J., and Alexander A. Stepanov. "Dynamic path planning for a mobile automaton with limited information on the environment." *Automatic Control, IEEE Transactions* on 31, no. 11 (1986): 1058–1063.

[3] Kareti, Srikumar, Weimin Shi, and S. Sitharama Iyengar. *Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms*. Oak Ridge, TN: Oak Ridge National Laboratory, 1993.

[4] Blum, Avrim, Prabhakar Raghavan, and Baruch Schieber. "Navigating in unfamiliar geometric terrain." In Proceedings of the *twenty-third annual ACM symposium on Theory of computing*, pp. 494–504. ACM, 1991.

[5] Afyouni, Imad, Cyril Ray, and Christophe Claramunt. "Spatial models for context-aware indoor navigation systems: A survey." *Journal of Spatial Information Science* 4 (2014): 85–123.

[6] Manikas, Theodore W., Kaveh Ashenayi, and Roger L. Wainwright. "Genetic algorithms for autonomous robot navigation." *Instrumentation & Measurement Magazine, IEEE* 10, no. 6 (2007): 26–31.

[7] Lumelsky, Vladimir J., and Alexander A. Stepanov. "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape." *Algorithmica* 2, no. 1–4 (1987): 403–430.

[8] Lumelsky, Vladimir J. "Algorithmic and complexity issues of robot motion in an uncertain environment." *Journal of Complexity* 3, no. 2 (1987): 146–182.

[9] Motlagh, Omid, Danial Nakhaeinia, Sai Hong Tang, Babak Karasfi, and Weria Khaksar. "Automatic navigation of mobile robots in unknown environments." *Neural Computing and Applications* (2012): 1–13.

[10] Ando, Hideki, Yoshinobu Oasa, Ichiro Suzuki, and Masafumi Yamashita. "Distributed memoryless point convergence algorithm for mobile robots with limited visibility." *Robotics and Automation, IEEE Transactions* on 15, no. 5 (1999): 818–828.

[11] Baeza-Yates, R.A., Culberson, J.C. and Rawlins, G.J. Searching in the plane. *Information and Computation*, 1991.

[12] Lewis, Jeremy S., and J. O'Kane. "Guaranteed navigation with an unreliable blind robot." In *Robotics and Automation (ICRA), 2010 IEEE International Conference*, pp. 5519–5524. IEEE, 2010.

[13] Zhu, David, and J-C. Latombe. "New heuristic algorithms for efficient hierarchical path planning." *Robotics and Automation, IEEE Transactions* on 7, no. 1 (1991): 9–20.

[14] Hwang, Yong K., and Narendra Ahuja. "A potential field approach to path planning." *Robotics and Automation, IEEE Transactions* on 8, no. 1 (1992): 23–32.

[15] Oriolo, Giuseppe, Giovanni Ulivi, and Marilena Vendittelli. "Real-time map building and navigation for autonomous robots in unknown environments." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions* on 28, no. 3 (1998): 316–333.

[16] Chatterjee, Ranajit, and Fumitoshi Matsuno. "Use of single side reflex for autonomous navigation of mobile robots in unknown environments." *Robotics and Autonomous Systems* 35, no. 2 (2001): 77–96.

[17] Wong, Sylvia C., and Bruce A. MacDonald. "A topological coverage algorithm for mobile robots." In *Intelligent Robots and Systems*, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 2, pp. 1685–1690. IEEE, 2003.

[18] Lumelsky, Vladimir J. "Dynamic path planning for a planar articulated robot arm moving amidst unknown obstacles." *Automatica* 23, no. 5 (1987): 551–570.

[19] Blake, A., M. Brady, R. Cipolla, Z. Xie, and A. Zisserman. "Visual navigation around curved obstacles." In *Robotics and Automation*, 1991. Proceedings., 1991 IEEE International Conference on, pp. 2490–2495. IEEE, 1991.

[20] Sutherland, I. A method for solving arbitrary wall mazes by computer. *IEEE transactions on Computers*, C–18(12): 1092 1097 1969.

[21] Lumelsky, V., Skewis, T. Incorporating range sensing in the robot navigation function. *IEEE Transactions on Systems, Man and Cybernetics*, 20(5): 1058, 1069, 1990.

[22] Koenig, Sven, Craig Tovey, and Yuri Smirnov. "Performance bounds for planning in unknown terrain." *Artificial Intelligence* 147, no. 1 (2003): 253–279.

[23] Canny, J. *Complexity of Robot Motion Planning*. MIT Press, Cambridge, M.A., 1987.

[24] Hopcroft, J., Joseph, D., Whitesides, S. *Planning, geometry and Complexity of Robot Motion*. iAblex Publishing Co., Norwood, New Jersy, 1987.

Appendix I: Construction Code in Mathematica

```
(*  We place circles that rest on the x-axis.
    The target is on the positive x-axis.
    Set the initial circle circ(x, r=y). *)
c = circ[0., 1.];
(*  Set the distance of the target *)
t = 2000

(*  This translates our internal circle to a graphics
    primitive that can be drawn *)
showCircle[circ[x_, r_]] := Disk[{x, r}, r]

(*  This rotates a point P clockwise around a point O
    with an angle theta. *)
rotate[P : {_, _}, O : {_, _}, theta_] :=
  {{Cos[theta], Sin[theta]},{-Sin[theta], Cos[theta]}}.(P - O) + O

(*  Find the point where the robot starts to follow a circle:
    If O is the origin of a circle Gamma with Ox < Tx, this finds
    the intersection of OT with Gamma that is furthest away from T.
    We obtain this point by taking the point Q where the x-axis
    touches Gamma. We can then rotate clockwise by 90deg + angle OTQ
    to get the point we are looking for *)
sp[Gamma_, Tx_] := Module[{T, x, y, r, angle, Q},
  T = {Tx, 0};
  {x, y} = List @@ Gamma; r = y;
  angle = ArcTan[Tx - x, y];
  Q = {x, 0};
  rotate[Q, List @@ Gamma, 90 Degree + angle]
];

(*  Similarly, from symmetry considerations, we can find the
    point where the robot leave the circle again *)
lp[Gamma_, Tx_] := Module[{T, x, y, r, angle, Q},
  T = {Tx, 0};
  {x, y} = List @@ Gamma; r = y;
  angle = ArcTan[Tx - x, y];
  Q = {x, 0};
  rotate[Q, List @@ Gamma, 180 Degree + 2 angle]
];

(*  Similarly again, this returns the circle arc that was
    followed on a circle (we have to define it counter-
    clockwise, now) *)
arc[Gamma_, Tx_] := Module[{T, x, y, r, angle},
```

```
  T = {Tx, 0};
  {x, y} = List @@ Gamma; r = y;
  angle = ArcTan[Tx - x, y];
  Circle[{x, y}, r, {180 Degree - angle, 90 Degree - 2 angle}]
];

(*  The length of the arc is *)
arclength[Circle[O : {_, _}, r_, {alpha0_, alpha1_}]] :=
  r Abs[alpha0 - alpha1]

(*  And the length of a line *)
linelength[Line[{a : {_, _}, b : {_, _}}]] := Norm[a - b]

(*  This finds the next circle based on the previous *)
nc[Null, Tx_] := Null;
nc[prev_, Tx_] := Module[{Px, Py, Lx, Ly, Ox, Oy, OxA, OyA, discr},
  {Px, Py} = List @@ prev;
  {Lx, Ly} = lp[prev, Tx];
  discr = Ly^3 Py (Lx Px + Ly Py - Lx Tx - Px Tx + Tx^2);
  If[discr >= 0,
    Ox = (Lx Ly Px + 2 Ly^2 Py - Ly Px Tx - 2 Sqrt[discr])/
    (Lx Ly - Ly Tx);
    OxA = (Lx Ly Px + 2 Ly^2 Py - Ly Px Tx + 2 Sqrt[discr])/
    (Lx Ly - Ly Tx);
    Oy = (Ly (Ox - Tx))/(Lx - Tx); OyA = (Ly (OxA - Tx))/(Lx - Tx);
    If[Oy > 0 && Abs[Tx - Ox] <= Abs[Tx - Px],
        circ[Ox, Oy],
        Null  ],
    (* else *)
    Null
  ]
];

(* Do the actual calculations *)
circles = NestList[nc[#, t] &, c, 100];
circles = DeleteCases[circles, Null];
startings = sp[#, t] & /@ circles;
leavings = lp[#, t] & /@ circles;
arcs = arc[#, t] & /@ circles;
pieces = Table[
  Line[{leavings\\[ii\\], startings\\[ii + 1\\]}],
  {ii, 1, Length[startings] - 1}
];
Opt = N[t + \[Pi]/2];
R = Total[arclength /@ arcs] + Total[linelength /@ pieces];
ratio = R/Opt
```

Acknowledgements

I am very grateful to my supervisor, Henk Meijer, for guiding me through this project. Working on a project with an open end and no guaranteed results is hard, and all our weekly meetings have inspired me again and again to try a different angle when I was stuck, or to focus when necessary. Doing this project with Henk has been very instructive, but above all, lots of fun.