# Landmark-Based Representations for Navigating Holonomic Soccer Robots

Daniel Beck, Alexander Ferrein, and Gerhard Lakemeyer

Knowledge-based Systems Group
Computer Science Department
RWTH Aachen
Aachen, Germany
{dbeck,ferrein,gerhard}@cs.rwth-aachen.de

**Abstract.** For navigating mobile robots the central problems of path planning and collision avoidance have to be solved. In this paper we propose a method to solve the (local) path planning problem in a reactive fashion given a landmark-based representation of the environment. The perceived obstacles define a point set for a Delaunay tessellation based on which a traversal graph containing possible paths to the target position is constructed. By applying A* we find a short and safe path through the obstacles. Although the traversal graph is recomputed in every iteration in order to achieve a high degree of reactivity the method guarantees stable paths in a static environment; oscillating behavior known from other local methods is precluded. This method has been successfully implemented on our Middle-size robots.

## 1 Introduction

One fundamental problem which has to be addressed for a mobile robot is the problem of navigation and path planning while avoiding to collide with obstacles in the environment. In a dynamic environment a navigation scheme with the ability to rapidly incorporate changes in the environment into the navigation process is required to safely and quickly navigate to the given target location. The navigation problem is often divided into two problems, which are solved independently from each other: *path planning* and *collision avoidance*. Whereas collision avoidance is regarded as a local problem, the path planning problem is often examined at a more global scale. In contrast to local methods which solely plan on the basis of the information gained from the current sensor readings, global methods rely on a global map of the environment, say, a floor plan of a building. Usually, those maps cover an area that exceeds the limited perception range of the sensors by far. As a consequence thereof the path can be planned from the start to the target position and not only around the next obstacles as it is the case for the local setting. Due to this fact, local methods often suffer from oscillating behavior. Drawbacks of global methods are that they do not account for avoiding obstacles and need a global map of the environment. Examples of global path planning are methods using probabilistic road maps [1] or dynamic

L. Iocchi et al. (Eds.): RoboCup 2008, LNAI 5399, pp. 25–36, 2009.

programming-based approaches [2]; examples of local path planning (collision avoidance) are curvature-velocity methods [3] or potential fields [4,5].

In this paper we propose a method which combines features from both, local as well as global path planning. The obstacles around the robot as they are perceived by means of its sensors are entered into a local map. As such, our method is local. We employ a geometric, landmark-based representation of the environment as opposed to many collision avoidance algorithms which are density-based and make use of a grid map for the perceived obstacles. Based on a Delaunay tessellation over the obstacles we construct a traversal graph. This traversal graph is a topological representation of all paths through the perceived obstacles to the target position. On this space we apply A* for finding a short and safe path. The safety of a path is accounted for by incorporating the distances between obstacles passed along the path into A*'s cost function.

This way we obtain a path from the current position of the robot, around the detected obstacles to the target position, i.e., the search cannot get trapped in certain obstacle configurations—potential field methods, for example, get stuck in local minima. Nevertheless, we reach a degree of reactivity that is comparable to other collision avoidance schemes, which is due to the fact that the traversal graph is re-computed and a new path is searched for after every sensor update. This is feasible since the traversal graph can be efficiently constructed and its size is linear in the number of obstacles. Furthermore, it is shown that the recomputation does not lead to an oscillating behavior but yields stable paths in a static environment. These characteristics make this a very well-suited approach for robotic soccer applications where the number of obstacles is rather small and new obstacle information extracted from camera images usually arrive with 20 – 30 Hz. Since holonomic robot platforms are the de-facto standard in the Middle-size league we describe how a path through the traversal graph as it is found by A* can be realized with such a robot. This means that, though path planning is involved, we keep up the reactivity of a local collision avoidance method, just like with potential fields. We show that the landmark-based representation is much more suited to scenarios like RoboCup and yield much smaller search problems. We compare the size of the path planning problems for landmark-based and density-based approaches and show that the branching factor as well as the solution length are smaller by an order of magnitude.

This paper is organized as follows. In Sect. 2 we discuss the large body of related approaches to the collision avoidance and path planning problem of mobile robots. In Sect. 3 we show our method in detail. We start with a concise description of the construction of the traversal graph and how A* is applied on the traversal graph to find a short and safe path. We show the optimality of the calculated path and that it is stable. Then, we discuss in detail how a Bézier curve is constructed based on the previously calculated path, before we show how we combine this approach with a potential field method to avoid nearby obstacles. Sect. 4 shows experimental results of our method. We conclude with Sect. 5.

## 2 Related Work

The problems of path planning and collision avoidance are essential for mobile robotics and the techniques developed to solve those problems are also of great importance for a multitude of applications of industrial robots, for animating digital actors, and also for studying the problem of protein folding. Therefore, a rich body of work exists in this area. A comprehensive overview of the subject is given in the excellent textbooks [6] and [7].

Since we developed our approach with the robot soccer scenario in mind and implemented it on our Middle-size league robots we concentrate the discussion of related work on the robot soccer domain and, additionally, also some other landmark-based approaches which are related to ours. In the robot soccer domain, most of the methods rely on density-based represenations; the most popular kind of such representations are occupancy grid maps. For example, potential field methods [4] make, in general, use this kind of representation. At sample points, scattered over the environment, a force is computed which is the combination of the repelling forces of the obstacles and the attracting force of the target. The idea is that the resulting force then guides the robot to the target. In [8] an in-depth discussion of the limitations of potential field methods is presented. A major drawback is the problem of getting stuck in local minima. In our approach we compute a path to the target position and, therefore, are not affected by such problems. Nevertheless, potential field-based methods appeal due to their simplicity and high degree of reactivity which is achieved by constantly re-computing the forces. Various extensions to the basic potential field approach have been proposed. For instance, in [9], a collision avoidance method is presented which combines artificial potential fields with a simulated annealing strategy to circumvent the problem of getting stuck in local minima. A similar approach is followed in [10] and [11].

Other grid based methods rely on search alogrithms to find a path to the target as it was presented in [12] for path planning with a humanoid robot on 2.5 D grid and in [13]. In [14] the authors define a cellular automaton on top of the grid representation. Similar to value iteration approaches the cellular automaton computes a lowest cost path to the target. Especially in sparsely populated environments a uniform resolution of the grid cell size, as it is commonly used, provokes unnecessary computations since the path is computed with the same precision regardless of the size of the free space around the robot. A (partial) remedy for this problem was proposed in [13] where the resolution of the grid decreases with the distance to the robot.

Generally, landmark-based approaches are better suited for sparsely populated environments because the length of the path is dependent on the number of obstacles and not on the resolution of the grid. Although, sensor-based variants of those approaches exist they are mostly intended for scenarios where a global map is available. Besides probabilistic roadmap methods, landmark-based and geometrical methods have in commmon that they rely on some kind of cell decomposition to sub-divide the free space into faces. One such example is given in [15]. There, the Voronoi diagram around the obstacles is constructed and a

path is defined as a sequence of edges in the Voronoi diagram. The Voronoi diagram is dual to the Delaunay tessellation and as such the approach mentioned above is similar to ours. With our approach, however, a trade-off between the riskiness of a path and its length can be implemented in a natural way which is not possible for Voronoi based approaches. This is because the obstacles are not part of the Voronoi diagram and in order to obtain the distances between the obstacles their position needs to be reconstructed from the given Voronoi diagram.

## 3 Triangulation-Based Path Planning

We now describe our landmark-based traversal graph in detail. With this representation and the cost function we use for planning, we found a good trade-off between shortness and the safety of the path. We prove that, for every point on the path, the calculated optimal path is stable given that the configuration of obstacles stays unchanged. Further, we show how a suchlike calculated path can be realized on a holonomic robot platform. In order to achieve the greatest possible reactivity, we moreover check if obstacles are very close to the robot and calculate, similar to the potential field methods, an avoidance course.

### 3.1 The Triangulation and the Traversal Graph

For the rest of the paper we assume that the robot is equipped with sensors that allows it to detect obstacles located in a circular perception field area around the robot using, say, an omnidirectional camera. The perceived obstacles are then represented in one of the following ways:

- Smaller obstacles (e.g., other robots) are represented by a point $P$ denoting the obstacle's center and an associated radius $r$ indicating the equi-directional extent of the obstacle. We write $|| P || = r$ to denote the extension of obstacle $P$ to be $r$.
- Obstacles which do not have an equi-directional extent (e.g., walls) are represented by a set of points and edges connecting the perceived contour points. Such points have no associated extent and the edges are explicitly marked as *not passable*. Generally, only the front-side contour can be perceived since no information about the extent of the obstacles on its rear side is available.

We construct a Delaunay tessellation for all such points which are perceived as obstacles. (See Fig. 1(a) for an example.)

**Definition 1 (Delaunay Tessellation).** *The Delaunay tessellation $DT(S)$ of a set $S$ of points in the plane is a graph with the vertices $V_{DT(S)}$ and the edges $E_{DT(S)}$. It is obtained by connecting any two points $p, q \in S$ with a straight line segment such that a circle $C$ exists which passes through $p$ and $q$ but no other point of $S$ lies within the circle $C$.*

A face $f^{(i)}$ of the tessellation is defined by the three vertices $v_0^{(i)}$, $v_1^{(i)}$, $v_2^{(i)}$. The edge $e_j^{(i)}$ connects the vertices $v_{j-1}^{(i)}$ and $v_{j+1}^{(i)}$ (indices mod 3). A concise overview

of Delaunay tessellations can be found in [16]. Due to the circumcircle criterion the triangulation as it is given by $DT(S)$ ensures that if the robot is located in the triangle $f^{(i)}$ the obstacles represented by $v_0^{(i)}$, $v_1^{(i)}$, $v_2^{(i)}$ are the next obstacles surrounding the robot. This implies that whenever the robot passes through any two obstacles it also crosses the edge in the triangulation $DT(S)$ between the vertices representing these two obstacles. Clearly, the safest way to pass through two obstacles is to drive right through the middle between them. This is why we choose the midpoints of the triangulation edges as way points. Then, the edges between the midpoints of a particular triangle represent, in a topological way, all possibilities to traverse the triangle in question. With this in mind, the problem of path planning now can be defined as finding a short and safe path through $DT(S)$ along those midpoints leading from the current position of the robot to the given target position. We thus construct a traversal graph $T(S)$ based on Delaunay tessellation $DT(S)$ over the set of points $S$ representing the obstacles.

**Algorithm 1 (Traversal Graph).** *Let $S$ be a set of points in the plane and $DT(S) = \langle V_{DT(S)}, E_{DT(S)} \rangle$ a Delaunay tessellation of $S$ (as shown in Fig. 1(a)) with $F = \{f^{(0)}, \ldots, f^{(m)}\}$ the set of faces of $DT(S)$ and $v_0^{(i)}$, $v_1^{(i)}$, $v_2^{(i)}$ the vertices of a face $f^{(i)}$ and $e_0^{(i)}$, $e_1^{(i)}$, and $e_2^{(i)}$ its edges as above. Let $R, T$ be points in the plane denoting the robot's position and the target position, respectively. Let $T(S) = \langle V_{T(S)}, E_{T(S)} \rangle$ be the traversal graph of $DT(S)$. $V_{T(S)}$ and $E_{T(S)}$ are constructed as follows:*

1. *Add the midpoints of all edges in $E_{DT(S)}$ to $V_{T(S)}$. For all triangles $f^{(i)}$ in $DT(S)$ that neither contain $R$ nor $T$ add the edges connecting the midpoints of $f^{(i)}$'s edges to $E_{T(S)}$. In case, $R$ or $T$ are located inside $f^{(i)}$ add further edges between the respective point and the midpoints of the circumjacent triangulation edges $E_{T(S)}$. (See Fig. 1(b).)*
2. *If neither $R$ nor $T$ are enclosed by $DT(S)$, we extend it in the following way:*
   (a) *determine the set of outer edges of $DT(S)$, i.e., the edges in $E_{DT(S)}$ that are not shared by any two adjacent triangles;*
   (b) *consider all outer edges (cf. arrows in Fig. 1(b)) for which no direct connection between the respective midpoint and $R$ or $T$ exists. (A direct connection exists if the straight line segment between the midpoint and $R$ or $T$ does not intersect with another outer edge);*
   (c) *for each such outer edge $e$ define line segments $s_{v_1}, s_{v_2}$ starting in $v_1, v_2$, resp., which are incident to $e$. The $s_{v_i}$ are angle bisectors of the outer angles between $e$ and the neighboring outer edges and are of length $|\, s_{v_i} \,| = 2 \cdot (||\, v_i \,|| + ||\, R \,||)$. Construct three further line segments $h_{v_1}$, $h_{v_2}$, and $h_m$, all perpendicular to $e$. The $h_{v_1}, h_{v_2}$ start in $v_1, v_2$, resp. ; $h_m$ starts in the midpoint $m$ of edge $e$; $|\, h_{v_i} \,| = |\, s_{v_i} \,|$ and $|\, h_m \,| = max\{|\, h_{v_i} \,|\}$. Fig. 1(c) illustrates this step;*
   (d) *let $S'$ be $S$ extended by the endpoints of these line segments. Re-establish $DT(S')$ and $T(S')$. Fig. 1(d) shows the resulting graph.*

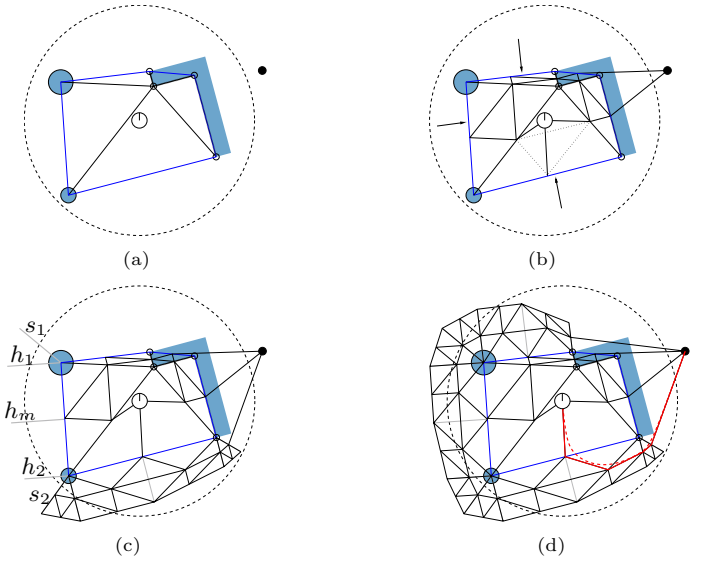



**Fig. 1.** An illustration of the construction of the traversal graph. The dashed circle around the robot depicts the perception border of the robot's sensors; the black dot on the right side is the target. The contour of non-circular obstacles is modeled by subdividing it into linear pieces which are represented by edges marked as not passable (cf. L-shaped obstacle to the right).

Every edge in $E_{T(S)}$ describes one possibility to traverse a triangle in $DT(S)$ by moving from the midpoint between two obstacles to the safest point to pass through the next two obstacles on the path. In order to also take into account paths that leave the area which is covered by the triangulation $DT(S)$ over an edge from which no direct path to the target exists, a construction as it is given in step 2 of the above construction algorithm is required. This construction guarantees that, topologically speaking, all possible paths through and also around the perceived obstacles can be represented. The traversal graph can be efficiently established. Note that the Delaunay tessellation can be established in $\mathcal{O}(n \log n)$ with $n = |S|$ (cf. [16]). The number of faces of $DT(S)$ and, consequently, also the size of $V_{T(S)}$ and $E_{T(S)}$ are in $\mathcal{O}(n)$. It can be shown that the whole algorithm runs in $\mathcal{O}(n)$.

### 3.2 Searching for a Short and Safe Path

The task is now to find a path between the robot $R$ and the target $T$ in the traversal graph $T(S)$. To find such a path, we use A$^*$ for two reasons: (1) it is known to be optimal efficient with an admissible heuristic, and (2) even though the worst-case complexity is $\mathcal{O}(b^d)$, where $b$ denotes the branching factor and $d$ the depth of the search tree, we have in our case a branching factor of at most 2. To apply A$^*$ we have to define an admissible heuristic and a cost function.

With the cost function we can influence the optimality criteria and furthermore prohibit edges with too few clearance.

**Definition 2 (Heuristic and Cost Function).** *For a Delaunay tessellation* $DT(S) = \langle V_{DT(S)}, E_{DT(S)} \rangle$ *and its traversal graph* $T(S) = \langle V_{T(S)}, E_{T(S)} \rangle$ *for a set* $S$ *of points in the plane,* $R$ *the robot's position, and* $T$ *the position of the target point, define a heuristic function* $h : V_{T(S)} \rightarrow \mathbb{R}$ *as* $h(p) = dist(p, T)$ *as the straight line Euclidean distance between a vertex* $p$ *in* $T(S)$ *and the target position* $T$*. The cost function* $g : V_{T(S)} \times \cdots \times V_{T(S)} \rightarrow \mathbb{R}$ *of a path* $p_0, \ldots, p_n$ *is defined as:*

$$g(p_0, \ldots, p_n) = \sum_{i=0}^{n-1} dist(p_i, p_{i+1}) + \sum_{j=1}^{n-1} clear(p_j)$$

*with* $clear(p_i) = \alpha \cdot ||e_i||^{-1} + \pi$*, where* $||e_i||$ *denotes the length of the triangulation edge* $e_i \in E_{DT(S)}$ *on which* $p_i \in V_{T(S)}$ *is located, and* $\pi$ *is a term punishing edges which are too short for the robot to cross.* $\pi$ *is defined as*

$$\pi = \begin{cases} \infty, \textit{ if } ||e_i|| < ||R|| + ||v_{i-1}|| + ||v_{i+1}|| \\ \quad \textit{or if } e_i \textit{ is marked as not passable} \\ 0, \quad \textit{otherwise} \end{cases}$$

*with* $||R||$ *denoting the robot's diameter,* $||v_{i-1}||$ *and* $||v_{i+1}||$ *the extent of the obstacles* $v_{i-1}, v_{i+1} \in V_{DT(S)}$ *incident to* $e_i$*.*

The result of the search will be a short and safe path through the graph that leads form the current position of the robot to the given target position.

**Theorem 1.** *Let* $S$ *be a set of points in the plane, let* $DT(S) = \langle V_{DT(S)}, E_{DT(S)} \rangle$ *be a Delaunay tessellation of* $S$ *with* $T(S) = \langle V_{T(S)}, E_{T(S)} \rangle$ *its traversal graph as constructed by Alg. 1. Let* $P^* = \langle p_0, \ldots, p_n \rangle$ *be the optimal path from* $p_0$ *to* $p_n$ *in* $T(S)$ *according to the cost function* $f = h + g$ *as given in Def. 2, and let* $R$ *be a point on the edge* $\langle p_i, p_{i+1} \rangle$ *of* $P^*$*. Let* $\tilde{P} = \langle p_{i+1}, \ldots, p_n \rangle \subset P^*$ *and* $P = \left\langle R, \tilde{P} \right\rangle$ *be a path starting in* $R$ *and following the optimal way points thereafter. Then, there exists no path starting in* $R$ *with lower costs than* $P$*.*

*Proof.* Assume the opposite, i.e., there exists a path $\tilde{P}' = \langle p'_{i+1}, \ldots, p'_m \rangle$ with $p'_m = p_n$ as depicted in Fig. 2 such that $P' = \left\langle R, \tilde{P}' \right\rangle$ is the optimal path from $R$ to the target position, but $P' \not\subset P^*$. Since, for all points on the edges of the path $\langle p_0, \ldots, p_i \rangle$, a path continuing with $\langle p_{i+1}, \ldots, p_n \rangle$ after $p_i$ has been chosen instead of one continuing with $\langle p'_{i+1}, \ldots, p'_m \rangle$, it must hold that: $g(p_i, \ldots, p_n) < g(p_i, p'_{i+1}, \ldots, p'_m)$. Under the assumption that the optimal path starting in $R$ follows the way points $\langle p'_{i+1}, \ldots p'_m \rangle$, it must hold that $g(p_i, p'_{i+1}, \ldots, p'_m) - (d' - d'_\epsilon) < g(p_i, \ldots, p_n) - \epsilon$ because otherwise $\langle p'_{i+1}, \ldots, p'_m \rangle$ would not have been selected as the optimal path from $R$ to the target. Since it holds for every triangle that each side of the triangle is longer than the absolute difference of the lengths of the other two sides, $d' - d'_\epsilon$ has to be less than $\epsilon$. This is in contradiction to the initial assumption, hence the claim follows. □

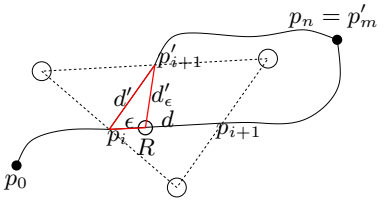



**Fig. 2.** Sketch of the proof of Theorem 1

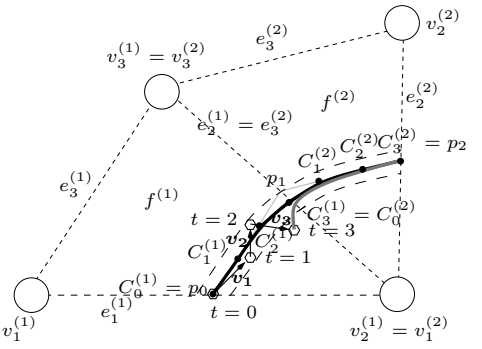



**Fig. 3.** Bézier curve of the path

**Corollary 1.** *The cost function of Def. 2 yields stable paths in* $T(S)$.

The corollary states that the decision which path to take does not oscillate for any point on the optimal path in the traversal graph $T(S)$. This makes the traversal graph a well-suited representation for the reactive path planning problem.

### 3.3 Realizing a Path on a Robot

After having found a path in $T(S)$, we have to realize this path on the robot. Let $P = \langle p_0, p_1, \ldots, p_{n-1}, p_n \rangle$ where $p_0 = R$ denotes the robot's location, the $p_i$ denote the points on the path, and $p_n = T$ the target position. As the path given through the path points is very angled, we aim at smoothing it. To this end, we approximate the path by a set of cubic Bézier patches. The parametrized representation of a Bézier curve of degree $n$ is given by $C(\tau) = \sum_{i=0}^{n} \binom{n}{i}(1-\tau)^{n-i}\tau^i C_i$, where $0 \leq \tau \leq 1$ is the curve parameter, and the $C_i$ are the control points of the curve. For our path approximation we make use of cubic Bézier patches with $n = 3$. Fig. 3 shows an example. The path through the faces $f_1$ and $f_2$ is given by the points $p_0$, $p_1$, and $p_2$ from which the control points $C_j^{(i)}$, $0 \leq j \leq 3$, $1 \leq i \leq 2$, of the two Bézier patches, can be constructed. The construction ensures that the curve consisting of several Bézier patches is $C^2$ continuous at the junctures. This property is very helpful for planning the robot's velocity. A detailed overview can be found, for example, in [17].

Now, for demonstrating how the robot drives along this curve, assume that, at time $t = 0$ the robot, located in $C_0^{(1)}$, is given the path through the points $p_0, p_1, p_2$. With these path points it derives the black curve given by the control points $C_j^{(1)}$. The robot now derives its next driving command by adding the difference quotient of the curve point at the next time step to its own movement vector. At time $t = 1$ the robot advanced to the depicted position. Its movement is described by its movement vector $\boldsymbol{v_1}$, depicted in the figure. At this time step the robots checks if it diverted too much from the curve calculated at $t = 0$ due to slippage or other imprecisions in executing the previous motion command. The

dashed lines left and right to the curve depict how much the robot might divert from the curve. To construct the black curve at $t = 1$, we must derive the first control point $C_0^{(1)}$ from where the robot started, as the Bézier curve is uniquely defined by its control points. This can be done by subtracting movement vector $\boldsymbol{v_1}$ from the robot's current position. At $t = 2$, we again check whether our actual position is near the original curve, which was calculated based on $p_0, p_1, p_2$. Note that in order to regain $C_0^{(1)}$, we project back the vector sum $\boldsymbol{v_1} + \boldsymbol{v_2}$ to time $t = 0$. At $t = 3$ the robot's location diverted too much from the curve in our example. Consequently, we have to compute a new curve (depicted with the dark gray curve in Fig. 3) to reach path point $p_2$ in face $f_2$.

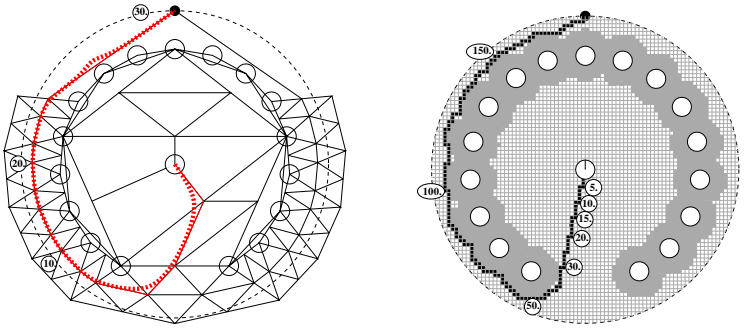
Generally, it is only necessary to project the robot's position back to the last juncture since only this point is necessary to compute the current Bézier patch, i.e., after passing $C_3^{(1)}$ we do not have to go back to $p_0 = C_0^{(1)}$.

### 3.4 The Fallback: Collision Avoidance with Nearby Obstacles

Although the aspect of the path's safeness was addressed during the search, it might nevertheless happen, under certain circumstances, that the robot gets too close to one of the obstacles. This might be due to the high dynamics of the environment (an obstacles approaches the robot very fast) or motion errors due to, say, slippage such that the robot ends up at another position than the intended one. To be on the safe side, we avoid bumping into the obstacle by using an additional reactive collision avoidance scheme which borrows fundamental ideas from the potential field method. If the robot's distance to an obstacle is less than a specified safety clearance, we compute a repelling force vector keeping the robot away from the obstacle and an attracting force vector guiding the robot to the next way point on the optimal path. These force vectors are computed in the same fashion as the force vectors in the potential field method. The intention behind this is to steer towards the next way point while increasing the distance between the robot and the obstacle in order to continue on the previously computed path. In contrast to a pure potential field approach not all detected obstacles have to be taken into account but only the closest. Furthermore, the drawback of potential fields, i.e., getting stuck in local minima, cannot occur here since it is guaranteed that no other obstacle is located in the area between the obstacle and the next way point. Thus, we do not run into the risk of oscillating motion behaviors.

## 4 Evaluation

In comparison to grid-based approaches a considerable speed-up in the search for an optimal path is achieved with our triangulation-based method. The reason for this lies, in general, in the lower number of triangles compared to the number of cells, which have to be searched. Moreover, the number of triangles depends on the number of obstacles and their relative positions to each other in contrast




(a) (b)

**Fig. 4.** Comparison of the traversal graph with a grid-based solution. Note: very small triangles are omitted in Fig. 4(a); length of path is correct, nevertheless.

to a non-adaptive sub-division of the area in cells with constant size.[1] Finally, the triangulation guides the search into the right direction in a natural way, i.e., it is not possible to stray too far from the optimal path.

In the following we give an example underlining this intuitive considerations. The example is a kind of a worst-case example which, nevertheless, is not unlikely to happen in reality. The situation is depicted in Fig. 4. The robot is surrounded by a number of obstacles arranged in an U-shaped fashion. The target is on the opposite side of the opening of the U such that the robot, at first, has to move away from the target to get out of the U-shape, and then needs to navigate around the obstacles to reach the target, eventually. For the following comparison we assume that perception radius is 4 m; the obstacles have a distance of 3 m to the robot's initial position; the robot has to keep a security distance of 50 cm to the obstacles; the cell size is $10 \times 10$ cm$^2$.

As can be seen in Fig. 4(a) the path from the initial position to the target traverses 30 triangles. The branching factor of the search tree is two (possible successors are the two other midpoints of the edges of the triangle the robot is entering) and consequently the size of the search tree is in $\mathcal{O}(2^{30})$. For the grid-based approach the search depth depends on the length of the path. The path leads around the obstacles which occupy the gray shaded cells. Note, the the security distance is added by the common method of obstacle growing, as depicted in Fig. 4(b). We now can estimate that the path, as it is depicted in black in the Fig. 4(b), traverses roughly 180 cells. Even if we assume that the branching factor is only four, i.e., no diagonal movements are allowed, the size of the search tree is in $\mathcal{O}(4^{180})$. Of course, these measures are both upper bounds which are hardly reached in practice—the effective branching factor of the search tree as it is generated by A$^*$ is in both cases smaller. Nevertheless, for the grid-based representation, nearly the whole inner of the U has to be searched before

[1] As described in Sect. 2, there also exist adaptive methods. Nevertheless, we compare our representation to non-adaptive grids as even an adaptive grid size does not remedy the basic problem of much longer paths.

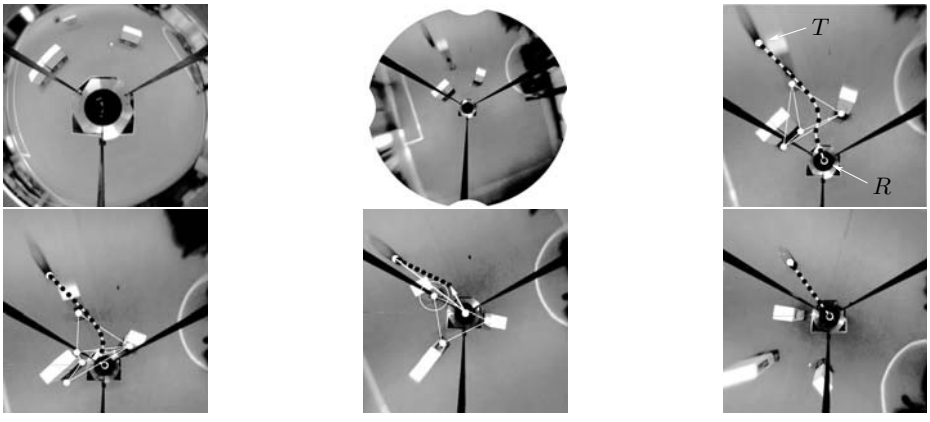



**Fig. 5.** (Top row from left to right): Omni-vision image of the scene, white boxes are the obstacles; undistorted image; the remaining images show a detailed view of the triangulation over the perceived obstacles, $R$ denotes the robot, $T$ the target. For the sake of clarity only the nearest obstacles are marked.

cells outside are taken into account. This shows that the search space is much larger than with our representation. Also, the length of the path gives a good impression of this fact.

Fig. 5 shows a real-world example. The top left image shows an omni-vision view of the scenario; the same view, though undistored this time, is shown in the middle. In the remaining images details of the undistorted views while navigating around the obstacles are depicted. The middle image in the bottom row pictures the nearby collision avoidance as it is mentioned in Sect. 3.4.

## 5 Conclusion

In this paper we presented a solution to the problem of reactive path planning. We introduce the traversal graph, a compact, landmark-based representation for sparsely populated environments like the robotic soccer domain. The traversal graph defines possible traversals of faces in a Delaunay tessellation leading through the midpoints of the Delaunay edges between two obstacles. In this graph we search for a path deploying A$^*$ with a heuristics and a cost function which yields a good trade-off between shortness and safety of the path. This path can be proved to be stable, that is, the optimal path does not change for any point on the path, under the assumption that the environment is static. This is a major improvement over local path planning methods like potential fields which cannot guarantee non-oscillating behaviors. Finally, we compare our method with a standard grid-based local path planning approaches and show that the solution can be computed much faster as the search problem is much smaller. This allows us to search for a new path each time, a new sensor update arrives.

The compactness of the representation as well as the stable path property make it possible to apply reactive path planning to the robot.

**Acknowlegments.** This work was partially supported by the German Science Foundation (DFG) in the Priority Program 1125 and by the Bonn-Aachen International Center for Information Technology (B-IT). We thank the anonymous reviewers for their comments.

## References

1. Canny, J.: Computing roadmaps of general semi-algebraic sets. The Computer Journal 36(5), 504–514 (1993)
2. Buhmann, J.M., Burgard, W., Cremers, A.B., Fox, D., Hofmann, T., Schneider, F.E., Strikos, J., Thrun, S.: The mobile robot rhino. AI Mag. 16(2), 31–38 (1995)
3. Simmons, R.: The curvature-velocity method for local obstacle avoidance. In: Proc. ICRA 1996. IEEE Computer Society Press, Los Alamitos (1996)
4. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. International Journal on Robotics Research 5(1), 90–98 (1986)
5. Borenstein, J., Koren, Y.: The vector field histogram - fast obstacle avoidance for mobile robots. IEEE Trans. on Robotics and Automation 3(7), 278–288 (1991)
6. LaValle, S.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
7. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)
8. Koren, Y., Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: Proc. ICRA 1991, pp. 1398–1404 (1991)
9. Park, M.G., Jeon, J.H., Lee, M.C.: Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In: Proc. ISIE 2001, vol. 3, pp. 1530–1535 (2001)
10. Zhang, P.Y., Lü, T.S., Song, L.B.: Soccer robot path planning based on the artificial potential field approach with simulated annealing. Robotica 22(5), 563–566 (2004)
11. Zhu, Q., Yan, Y., Xing, Z.: Robot path planning based on artificial potential field approach with simulated annealing. In: Proc. ISDA 2006, pp. 622–627 (2006)
12. Gutman, J., Fukuchi, M., Fujita, M.: Real-time path planning for humanoid robot navigation. In: Proc. IJCAI 2005, pp. 1232–1238 (2005)
13. Behnke, S.: Local multiresolution path planning. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS, vol. 3020, pp. 332–343. Springer, Heidelberg (2004)
14. Behring, C., Bracho, M., Castro, M., Moreno, J.A.: An algorithm for robot path planning with cellular automata. In: Proc. ACRI 2000, pp. 11–19. Springer, Heidelberg (2000)
15. Sahraei, A., Manzuri, M.T., Razvan, M.R., Tajfard, M., Khoshbakht, S.: Real-Time Trajectory Generation for Mobile Robots. In: Basili, R., Pazienza, M.T. (eds.) AI*IA 2007. LNCS (LNAI), vol. 4733, pp. 459–470. Springer, Heidelberg (2007)
16. Aurenhammer, F., Klein, R.: Voronoi diagrams. In: Sack, J.R., Urrutia, J. (eds.) Handbook of Computational Geometry. Elsevier Science Publishers, Amsterdam (2000)
17. Foley, J.D., Phillips, R.L., Hughes, J.F., van Dam, A., Feiner, S.K.: Introduction to Computer Graphics. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1994)