MOTION PLANNING AND CONTROL OF A CAR-LIKE ROBOT IN AN ENVIRONMENT CLUTTERED WITH STATIC OBSTACLES

BY

NARCIS GHIȚĂ and MARIUS KLOETZER

Abstract. This paper proposes a method for planning and controlling the motion of a car-like robot such that, by starting from a given initial position, a goal position is reached in an environment cluttered with static obstacles. The outcome of the planning part is an implementable trajectory guaranteeing a collision-free movement, and the control part consists from a trajectory following technique. The planning part iterates three main steps: first, an angular path linking the initial and the final position is found by using a cell decomposition of the environment. Second, a smooth trajectory satisfying the curvature constraints of the car is obtained. Third, we develop a procedure that takes into account the size of the car and tests if the smooth trajectory is feasible from the point of view of avoiding collisions with obstacles. The developed approach is supported by illustrative examples that use different cell decompositions.

Key words: motion planning, car-like robot, cell decomposition, path following.

2000 Mathematics Subject Classification: 93C85, 70E60.

1. Introduction

The goal of a classical navigation problem for a mobile robot consists from finding a control strategy such that the robot reaches a desired position, without colliding with obstacles from the environment [1], ..., [3]. There are two main research directions emerging from this area. Some researchers focus on robots with simple dynamics and negligible size and try to reduce the limited expressivity of the classical navigation task, by considering high-level, human-like specifications for the motion task [4], ..., [6]. Concomitantly, other researchers aim in solving the classical navigation problem for mobile robots
with complicated dynamics, such as nonholonomic or underactuated systems [7], [8].

The existing solutions for the classical navigation problem can be divided into three categories [9]: methods based on constructing global potential functions (navigation functions), constructing roadmaps for the free configuration space (e.g., visibility graphs or Voronoi diagrams), decomposing the free space into cells (geometric shapes) and creating a graph corresponding to the adjacency relation between cells. All these methods assume that the environment map is known and that the robot is reduced to a fully actuated point in the free configuration space. Moreover, some of these methods have some additional specific drawbacks, e.g., the potential functions can have local minima, the construction of navigation functions without local minima is in general very difficult, while the methods based on visibility graphs result in robot trajectories that are very close to obstacles.

An impressive amount of research on path planning for mobile robots focuses on simple robot dynamics, either fully actuated or underactuated with differential-wheel driven structure (close to unicycle). Although such approaches yield useful results, a more realistic approach from the point of view of existence of large scale mobile robots is to focus on car-like robots. The used math models are usually formulated either in a fixed frame of the environment, or in frames that moves along the desired path of the robot [9].

The goal of this paper is to develop an algorithmic method for constructing a solution to the navigation problem for a car-like robot. We assume that the car-like robot evolves in a planar and bounded environment, cluttered with static and convex polygonal obstacles. The complete map of the environment is assumed to be available.

The solution we propose includes the following main steps: first, by using a cell decomposition method, we find an angular path, consisting from a succession of line segments, which links the start position with the goal position of a specific reference point on the robot. Second, by taking into account the steering capabilities of the car-like robot, we construct a smooth path, consisting of line segments and arcs of circles. Third, by considering the physical dimensions of the robot and the smooth path it should traverse, we use a procedure that iterates the first two steps until the obtained smooth path is correct, in the sense that the robot can follow it and during the motion no obstacle is hit. Finally, by assuming that the position of the robot can be always read, we employ a feedback controller that has the obtained smooth path as a reference trajectory for the robot. The first three steps of our solution form the planning part, and they can be performed offline (before the actual movement). Thus, the computational load is reduced during the real-time control of the car-like robot.

As in [10], [11], we construct a solution to a navigation problem by using a cell decomposition method. However, our method mainly focuses on planning
the motion of car-like robots, while [10], [11] are interested in controller design, and their strategies are conservatively adapted to unicycle robots.

The construction of the smooth path from our approach seems related to constructing trajectories for car-like robots by using Dubins curves [12],[13],[15]. However, the focus in [12], [16] is on finding local implementable trajectories the robot can follow, while taking into account the initial and final orientation of the robot as well, but while ignoring any obstacle. We aim in generating an implementable trajectory that guarantees a collision-free movement between obstacles and that includes arcs of circle with different radii, while all arcs from Dubins curves have the same radius.

A path planning and control policy for nonholonomic systems are described in [7], [8] where the environment is no longer decompose in polygonal cells, but the boundary in the local coordinates is a smooth surface. In order to simplify the control, the dimension of environment is increased, by considering the third dimension as the orientation of the robot. Due to complexity issues, this path planning algorithm is mainly designed to work in environments with relatively few obstacles.

For exploring with car-like robots large environments where the map is unavailable before the experiment, [17], [18] propose an approach named framed quad-tree. Although in [17] the main problem consists in exploring the environment, some advantages of using quad-trees for path planning are stated.

In order to check if the robot does not collide with obstacles, we will use a slightly over approximated path derived from the basic construction proposed by [19]. Unlike [19], we initially generate a path for a specific point on the robot by using a cell decomposition method. An overview of feedback control techniques for path following car-like robots can be obtained from [20],[21],[22].

The remaining of this article is organized as follows. Section II presents some preliminary material, while Section III proposes a path generating algorithm for car-like robots. In Section IV we briefly discuss the control strategy allowing the motion along the generated path. The theoretical developments are illustrated in Section V by an example using trapezoidal decompositions. The final section comments on the importance of our work and possible future research directions.

2. Preliminaries

For the car-like system we consider the kinematic model given by equation (1):

$$
\begin{align*}
\dot{x} &= v\cos(\theta) \\
\dot{y} &= v\sin(\theta) \\
\dot{\theta} &= v\tan(\phi) / L
\end{align*}
$$
where \((x, y)\) are the Cartesian coordinates in a fixed frame \((S)\) of the reference point \(P_m\), located at mid-distance of the actuated wheels, angle \(\theta\) characterizes the robot’s chassis orientation with respect to frame \(S\), and \(L\) is the distance between the rear and front axle. The control inputs are \(v\), which is the vehicle’s velocity, ensured by the rear wheels, and \(\phi\), which is the vehicle’s steering wheel angle, due to the front wheels, and measured with respect to the current chassis orientation, as depicted in Fig. 1 a). We will assume a constant velocity \(v\), and a steering angle restricted by physical limits of the car-like robot, \(\phi \in [-\phi_{\text{max}}, \phi_{\text{max}}]\). For a detailed development of model (1) we refer the interested reader to [3], [23], while a discussion on when a kinematic model is sufficient and when it is necessary to use a dynamic model for a car-like robot is presented in [9].
The problem we solve in this paper can be formulated as:

Problem 1: Given a car-like robot with model (1), which can evolve in a bounded environment cluttered with convex polygonal obstacles, a start position \((x_0, y_0)\) and a final position \((x_f, y_f)\), find a control strategy that drives the robot to the goal (final) point while avoiding any collision with obstacles.

We assume that the initial orientation of the robot’s chassis can be chosen, while the final orientation is not of interest to us. Furthermore, we will assume that, as any obstacle, the environment is also bounded by a convex polygon, such an assumption being easily accomplished with a bit of conservatism for any arbitrarily-bounded environment. Also, we mention that the assumption of having convex obstacles is not restrictive, since any concave polygonal obstacle can be represented by some overlapping convex ones. We will denote the obstacles by \(O_1, \ldots, O_n\), and the environment by \(E\).

In solving Problem 1, we will use cell decomposition methods, which are shortly defined in the remainder of this section. A formal treatment of such methods would go beyond the scope of the current paper, and we mention that algorithms performing the mentioned decompositions are available.

A cell decomposition is a partition of the free space (the part of the environment not occupied by obstacles) into polygonal regions of the same type. Our approach can be used for cell decompositions where the cells are convex polygons, typical examples including trapezoidal cells [2], triangular cells [11], [24], polytopal cells [5], or rectangular cells resulted from a quadtree decomposition algorithm [10], [24]. By using a cell decomposition algorithm, the free space in which a point robot can move is abstracted to a finite graph \(G = (C, A)\), where each node \(C \in C\) corresponds to a cell, and the edges (arcs) from \(A\) correspond to adjacency among cells. With a slight abuse of notation, we will use \(C_i, i = 1, \ldots, |C|\), to denote either the label of a node from the graph, or the whole polygonal cell corresponding to that node. Furthermore, we assume that \(A\) has the form of a symmetric adjacency matrix, \(A \in \mathbb{R}^{|C| \times |C|}\), where means there is an arc linking \(C_i\) with \(C_j\), with weight (cost) \(A_{i,j}\), and \(A_{i,j} = 0\) otherwise.

Since the cells we are dealing with are bounded convex polygons, we mention that such a region can be described in two equivalent ways: as the convex hull of its vertices (usually the vertices result from a cell decomposition algorithm), or as the intersection of a set of closed half-planes (form that is used for example in LP problems). These two representations are denoted by V-representation and H-representation, respectively.
3. Path Generating

For solving Problem 1, we aim in generating a smooth trajectory accomplishing the imposed task. As described in Section I, this smooth trajectory will be obtained from an iterative procedure that first generates an angular trajectory, based on cell decomposition, than smoothen it and tests it against intersections with obstacles. Before describing the iterative procedure (subsection III.C), we will first focus on constructing an angular path (subsection III.A) and a smooth path (subsection III.B).

3.1. Angular Path

By performing a cell decomposition of the free space from environment $E$ (as mentioned in Section II), and we assume that a graph $G = (C, A)$ is obtained. For simplicity, we assume that all arcs have unitary cost, although some approaches might compute weights by considering the size of the car-like robot, or the energy spent for moving between adjacent cells. Next, we identify the cells including the start and the final positions, and we denote them by $C_0$ and $C_f$, respectively.

By performing a search on graph $G$, with start node $C_0$ and goal node $C_f$, we obtain a path showing the succession of cellular regions the robot should follow. Since we consider unitary costs on arcs of $G$, if the path was found by a minimum-cost search algorithm (e.g. Dijkstra), it includes the minimum number of cells. We simply construct the angular path ($P_{\text{ang}}$) by choosing a single point from each cell from the path in $G$, and by linking these points in the order the corresponding cells should be traversed. The point chosen from each cell can be either the centroid of the cell (e.g., for polytopal decompositions), or the centroid of the common line segment shared by the current cell with the next one, excepting, of course, the last cell from the path (e.g. in the case of trapezoidal and rectangular decompositions). Since we are dealing with convex cells, this choice of intermediary points guarantees that the angular path is contained in the sequence of regions defined by the cells from the path.

We finish the construction of the angular path by adding the starting point $(x_s, y_s)$ at the beginning of the path, and the ending point $(x_f, y_f)$ at the end. We denote the points defining the angular path $P_{\text{ang}}$ by $(p_i, p_2, \ldots, p_N)$, where $p_i = (x_i, y_i) \in \mathbb{R}^2$, $i = 1, \ldots, N$, and $p_1 = (x_s, y_s)$, $p_N = (x_f, y_f)$, $p_1, p_2 \in C_0$, $p_{N-1}, p_N \in C_f$. The approach we use for constructing $P_{\text{ang}}$ is common in navigation planning that uses cell decompositions, and we refer the interested reader to [2] for a more detailed treatment of the trapezoidal-cell case.
3.2. Smooth Path

As stated before, \( P_{\text{ang}} \) is the union of the line segments \([p_i, p_{i+1}]\), \( i = 1, \ldots, N-1 \), and can be described in Cartesian coordinates of the fixed frame \( S \) by:

\[
P_{\text{ang}} = \bigcup_{i=1}^{N-1} \{(x, y) \in [x_i, x_{i+1}] \times [y_i, y_{i+1}] | a_ix + b_i = y\}.
\]

\( P_{\text{ang}} \) is in general a non-smooth trajectory, because the line segments defining it have might have different slopes. Since the car-like described by (1) can only follow smooth trajectories, we aim to generate a smooth path \( P_{\text{smooth}} \) by starting from \( P_{\text{ang}} \). The idea of generating \( P_{\text{smooth}} \) resembles Dubins curves [12], which interconnect circle arcs and line segments such that a desired position and orientation are reached. Our approach differs in the following sense: we already have an angular path that gives us the orientation of the intermediary line segments, and we smoothen this path by using circle arcs with different radii.

In the following, we denote the orientation (slope) of each segment \([p_i, p_{i+1}]\), \( i = 1, \ldots, N-1 \), by \( \theta_i \), where \( \theta_i = \text{atan}(a_i) \). Let us assume that at a certain time moment the robot is along a line segment \([p_i, p_{i+1}]\) (its reference point is on \([p_i, p_{i+1}]\), and its orientation is \( \theta_i \)). For being able to move the robot such that it reaches any position from \([p_i, p_{i+1}]\) and orientation \( \theta_{i+1} \), we interconnect the line segments \([p_i, p_{i+1}]\) and \([p_{i+1}, p_{i+2}]\) by an arc of circle of radius \( R \), tangent to both segments, as illustrated by Fig. 2a. In choosing the value of \( R \) we make a tradeoff between the following two aspects: on one hand, the minimum value of \( R \) is imposed by the maximum steering angle \( \phi_{\text{max}} \), and such a minimum radius would keep \( P_{\text{smooth}} \) as close as possible to \( P_{\text{ang}} \). On the other hand, the smaller the radius is, the larger the discontinuity will be in control input \( \phi \), when steering the robot from a line segment to the arc of circle.

\( P_{\text{smooth}} \) is generated by iterating the above idea for all successive line segments from \( P_{\text{ang}} \). Due to space constraints, we do not include the algorithm used for generating \( P_{\text{smooth}} \). However, we mention that for some \( P_{\text{ang}} \) we cannot generate \( P_{\text{smooth}} \), simply because the minimum radius imposed by \( \phi_{\text{max}} \) might not be small enough to interconnect short line segments. In such a case, the algorithm used for generating \( P_{\text{smooth}} \) returns a set \( I \) of pairs of indices for line
segments from $P_{\text{ang}}$ that cannot be smoothly interconnected.

![Diagram](image)

Fig. 2 – $P_{\text{ang}}$ and $P_{\text{smooth}}$ described in section III A and III B (a); Polygonal over approximation of the area swept by the robot when following a circle arc (b).

### 3.3. Feasible Path

We say that an obtained $P_{\text{smooth}}$ is feasible if no obstacle is hit while the robot moves along it. In order to check the feasibility of $P_{\text{smooth}}$ found as in subsection III.B, we first consider that the initial orientation of the robot chassis is chosen to be $\theta_1$. Then, while the robot moves forward along the part of the line segment $[p_1, p_2]$ included in $P_{\text{smooth}}$ it sweeps a rectangle with a side equal to $l$. Checking the possible intersections of this rectangle with obstacles $O_1, \ldots, O_n$ reduces to checking the existence of a solution for a linear programming (LP) problem, where the constrained set is given by the polytopal obstacles and the rectangle swept by the robot, and a trivial cost is assumed. For checking possible intersections with obstacles while the robot follows a circle arc from $P_{\text{smooth}}$, we over approximate the area swept by the robot with a polyhedron, as shown in Fig. 2b, and then we solve a similar LP problem. Although this over approximation increases the conservativeness of our approach, solving an LP is computationally attractive than numerically solving a system of nonlinear equations with constrained solutions. The over approximation of the curved swept area is similar with the one used in [19], the difference consisting in the construction and the shape of the considered polyhedron.

The feasibility of $P_{\text{smooth}}$ is checked by iterating the above approach for all its line segments and circle arcs. When some elements of $P_{\text{smooth}}$ yield intersections with obstacles, their corresponding pairs of indices for line segments from $P_{\text{ang}}$ are stored in a set $I$, and the current path is unfeasible.

So far we have described all the individual procedures for constructing
Algorithm 1 – Find a feasible path
Perform cell decomposition and obtain \( G = (C, A) \)

\[
\text{WHILE TRUE}
\]

\[\text{Find } P_{\text{ang}} \text{ as described in subsection III.A}\]

\[\text{IF } P_{\text{ang}} \text{ not found}
\]

\[\text{break /* Problem 1 is unfeasible */}\]

\[\text{ENDIF}\]

\[\text{Construct } P_{\text{smooth}} \text{ from } P_{\text{ang}}\]

\[\text{Check feasibility of } P_{\text{smooth}}\]

\[\text{IF } P_{\text{smooth}} \text{ was not created or is not feasible, }
\]

\[A_{i,j} = 0, A_{j,i} = 0 \text{ for any } \{i,j\} \in I\]

\[\text{ELSE}
\]

\[\text{break/* } P_{\text{smooth}} \text{ is a feasible trajectory*/}\]

\[\text{ENDIF}\]

\[\text{ENDWHILE}\]

Algorithm 1 iteratively finds an angular path, a smooth trajectory in Section IV. Otherwise, links from set \( I \) (that prevented the construction of \( P_{\text{smooth}} \), or that induced unfeasibility of the current path) are removed from graph \( G \), and a new angular path is searched. Since we remove an arc from the graph \( G \) at each iterative step, Algorithm 1 is guaranteed to terminate in a finite number of steps (if a feasible path cannot be found, at some point \( P_{\text{ang}} \) does not exist and the procedure stops).

4. Control Strategy

The control strategy is based on a small variation linear model, obtained from the linearization of nonlinear system (1) around a generic point \((x, y, \theta)\) of the path \( P_{\text{smooth}} \) (meaning the two coordinates, supplemented with information about angle). The generic points are found by assuming a constant velocity \( v \) when moving along \( P_{\text{smooth}} \), and the angle is given by the tangent in the current coordinates at \( P_{\text{smooth}} \). The small variation linear model is further discretized by using a sampling period chosen in accordance with value of \( v \). The feedback law providing the path and tests the feasibility of the smooth path. If the current path is feasible, it will be used as reference control input \( \phi \) is designed by assigning eigenvalues able to ensure a reasonable decrease of the state variables (i.e., the variations from the nominal point). At each sampling time, the nominal point is
Narcis Ghiță and Marius Kloetzer

considered as the space origin for the discrete-time form of the linearized model.

5. Simulation Results

To illustrate the proposed planning strategy, we developed a simulation study in Matlab. The example we focused on is depicted in Fig. 3a, where a trapezoidal decomposition is performed. We have chosen this decomposition after a number of tests for trapezoidal, rectangular, triangular and polytopal cells. The advantage offered by the trapezoidal decomposition consists in the algorithm’s simplicity and the reduced number of resulting cells. For example, Fig. 3b represents a comparison term corresponding to the decomposition of the same environment by using rectangular cells resulted from a quad-tree approach. As a consequence, \( P_{\text{ang}} \) and \( P_{\text{smooth}} \) shown in Fig. 3a allow a robust implementation for the path following strategy. The advantage of the trapezoidal cells commented above does not reflect a general recipe, since for other environments, one can obtain trapezoids that are too thin.

![Fig. 3 – \( P_{\text{ang}} \) and \( P_{\text{smooth}} \) obtained with: a – trapezoidal cell decomposition; b – quad-tree cell decomposition.](image)

The generated \( P_{\text{smooth}} \) was used in the control strategy briefly described by Section IV, and the trace of the car-like robot motion is illustrated in Fig. 4a - for the trapezoidal decomposition and, respectively, in Fig. 4b – for the quad-tree approach. A comparative analysis of Fig. 4a and 4b points out a supplementary advantage (not commented above) of the path generated with trapezoidal cell decomposition. Generally speaking, by considering this path and any two neighbour obstacles with the path located in between, the points of the path are placed at approximately equal distances from the two obstacles. Thus, adequate lateral margins are ensured for the motion of the robot (whose
width cannot be neglected). This property is ensured by the construction procedure selecting the mid points of the edges, and it may not hold true for the quad-tree cell decomposition. Figs. 4a and 4b exhibit visible differences for the robot motion in the descending part of the two sigmoid-type “tubes” that keep the moving robot inside. In Fig. 4b the motion is very close to an obstacle (meaning an increased risk of collision), whereas in Fig. 4a the motion progresses at reasonable distances from all the obstacles (with low risk of collision). Actually the differences between Figs. 4a and 4b are predictable by examining Figs. 3a and 3b, since the angular path in Fig. 3b is not equally distanced from the adjacent obstacles.

![Figure 4](image)

**Fig. 4** – Robot motion in the environment when: a – the trapezoidal cell decomposition is used; b – the quad-tree cell decomposition is used.

6. Conclusions

This paper proposed a solution for planning and controlling a car-like robot in planar environments cluttered with obstacles. The novelty refers to the development of an iterative procedure that returns a smooth path based on cell decomposition methods and connections inspired by Dubins curves. Although the approach we propose is not complete, since a solution might not be found even if one exists, it has the advantage of low computational complexity and easy implementation.

The proposed solution deserves further investigations in extending the problem to more complex, human-like robot specifications.

**Acknowledgements.** The first author acknowledges the support of the EURODOC (Doctoral Scholarships for research performance at European level) project, financed by the European Social Fund and the Romanian Government. The second author acknowledges the support of the CNCSIS-UEFISCSU project PN-II-RU PD code 333/2010.
Both authors are indebted to Professor Octavian Păstrăvanu for useful discussions on various topics of this paper.

Received: May 28, 2010

“Gheorghe Asachi” Technical University of Iaşi, Department of Automatic Control and Applied Informatics
e-mails: narcis_ac@yahoo.com
marius_kloetzer@yahoo.com

REFERENCES

Lucrarea propune o metodă automată ce permite planificarea și controlul mișcării unui robot mobil într-un mediu care conține obstacole poligonale static. Robotul este considerat de tip mașină, iar scopul problemei este de a genera o strategie de control astfel încât, pornind dintr-o poziție inițială specificată, robotul să atingă o poziție dorită, evitând în timpul mișcării coliziunile cu obstacolele existente. Tehnica propusă constă în două părți principale: o parte de planificare, ce are drept rezultat o traiectorie implementabilă de robot care garantează satisfacerea problemei date, și o parte de control, care constă în urmărirea traiectoriei de referință generată anterior. Una din contribuțiile principale ale lucrării este procedura de planificare, care constă în iterarea a trei pași de bază până când o soluție este obținută, sau problema este considerată
nefezabilă. Primul pas constă în generarea unei traiectorii unghiulare ce unește punctul de start cu cel de stop, folosind o descompunere a spațiului liber de evoluție în celule convexe de același tip. În pasul următor, pe baza traiectoriei unghiulare este obținută o traiectorie netedă ce satisfac constrângerile de natură fizică ale robotului (raza minimă a curbelor ce pot fi realizate). Ultimul pas ia în considerare dimensiunile robotului și verifică dacă evoluția mașinii de-a lungul traiectoriei obținute în al doilea pas garantează evitarea obstacolelor. Lucrarea include exemple ilustrative ce susțin aplicabilitatea metodei propuse.