ꟼ periodica polytechnica

Electrical Engineering and Computer Science
57/3 (2013) 65–76
doi: 10.3311/PPee.2099
http://periodicapolytechnica.org/ee
Creative Commons Attribution

RESEARCH ARTICLE

Nonholonomic Path Planning for a Point Robot with Car-Like Kinematics

Domokos Kiss / *Gábor* Tevesz

Received 2013-04-29, revised 2013-06-24, accepted 2013-06-24

Abstract

A new approach of nonholonomic path planning for car-like robots is presented. The main idea is similar to many existing approaches which obtain a path in two phases. It is familiar in nonholonomic planning that at first a holonomic path is planned which is approximated by a nonholonomic one in a second step by subdividing it into smaller parts and replacing them with local paths fulfilling the kinematic constraints. These methods mostly rely on probabilistic methods and heuristic optimization. Our approach uses a holonomic preliminary path as well, but it serves only as a "loose guidance" to the second phase of the planning process. The final path is not required to contain any of the intermediate points of the preliminary path at all. The method is effective in environments consisting of narrow corridors but having wider free areas as well which can be used for maneuvering.<br />

Keywords

Path Planning · Nonholonomic Robots · Narrow Environments · Cell Decomposition

Acknowledgement

The work in the paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME". This project is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009

Domokos Kiss
Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Magyar tudósok krt. 2., H-1117 Budapest, Hungary
e-mail: domokos.kiss@aut.bme.hu

Gábor Tevesz
Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Magyar tudósok krt. 2., H-1117 Budapest, Hungary
e-mail: tevesz@aut.bme.hu

1 Introduction

Intelligent path planning and control of autonomous vehicles becomes nowadays increasingly widespread in industry and even in everyday life. Besides robotic cleaning machines autonomous park assist systems for passenger cars appeared on the market in the last years and unmanned aerial vehicles are flying with increasing autonomy. According to their popularity, research in this field is considerably active in the last decades [1–4].

The navigation task of autonomous mobile robots consists of more solvable subproblems, such as sensing and localization, environment mapping, path planning and motion control. Among mobile robots moving on a planar surface, the most popular locomotion systems are based on rolling wheels, according to their mechanical simplicity and popularity among everyday vehicles. From a control theoretic view, the rolling without slipping constraint of wheels induce *nonholonomic constraints*, which are nonintegrable differential constraints over the configuration space of the system. Nonholonomic constraints do not prohibit the controllability of a system, but they cause remarkable difficulties (this can be acknowledged by anyone who ever tried to parallel park a car). Although controlling a nonholonomic vehicle is a control theoretic challenge, it is worth incorporating the knowledge about the specific system in the path planning task as well. Planning infeasible paths for nonholonomic systems can baffle the whole navigation process.

This paper describes a path planning algorithm for robots with car-like kinematics moving on a horizontal planar surface. It is intended to solve navigation problems where the robot has to cross narrow corridors in order to reach the goal. Since car-like robots have a nonzero minimum turning radius, such environments require in some cases non-trivial maneuvering (including more reversals) between the narrowings. Our planning approach employs a geometric local planner based on the fact that in the absence of obstacles two configurations of a car-like robot can always be connected by a three-segment CCS or SCS subpath (where C stands for circular and S for straight path segments). A global preliminary (unconstrained) path is planned in advance based on cell decomposition of the free space, which is used to guide the local planner. This guidance means that the local planner looks for feasible subpaths along the preliminary path – without forcing the resulting path to have common intermediate points with it. The algorithm is currenty implemented for point-shaped robots (having a minimum turning radius constraint), but can be extended to vehicles of polygonal shape without theoretical complications.

The paper is organized as follows. In Section 2 the existing methods for nonholonomic planning are surveyed shortly. The next section states the problem covered by our approach. Section 4 describes the local planner module used repeatedly during the path planning process and Section 5 is about planning the preliminary guiding path. In Section 6 we summarize the whole planning process, investigate its computational complexity and demonstrate its effectiveness by simulation results. Section 7 draws the conclusions and gives directions of future work.

2 Related Work

Path planning is a widely studied research area of robotics [1–4]. Many different approaches can be found in the literature according to the different classes of systems. The simplest case is the pure geometric path planning for holonomic systems, which basically means choosing a function λ from the set Λ of continuous functions defined on the interval $[0,1]$ and taking their values from the free configuration space, connecting a given initial and final configuration q_I and q_G of the robot:

$$\Lambda = \{\lambda : [0,1] \to C \mid \lambda(s) \notin C_{obs}\ \forall\ s \in [0,1], \lambda(0) = q_I,\ \lambda(1) = q_G\} \quad (1)$$

where C is the configuration space of the robot and $C_{obs} \subset C$ is the set of configurations in collision. For a rigid robot moving in the plane, the configuration $q = [x,\ y,\ \theta]^T$ describes the position of the reference point in the plane and the orientation with respect to the vertical axis.

For problems including not only geometric constraints (i.e. obstacles) but kinematic constraints of the robot model as well, the above definition may not be sufficient to describe a solution path. A useful path planning algorithm for such systems has to deliver a solution which is *feasible* (i.e. it obeys the motion equation of the robot) and is safe with respect to obstacles.

2.1 Nonholonomic Systems

The vast majority of available planning methods deal with driftless control affine systems, which have the form

$$\dot{x} = \sum_{i=1}^{m} h_i(x)u_i \quad (2)$$

where the state $x \in X$ consists of the configuration variables and their derivatives, each h_i is a vector field on the state space X and $u = [u_1 \dots u_m]^T \in U$ is the vector of control variables. In many cases $x = q$, i.e. the configuration is treated as the state. The nonholonomy and controllability properties of a system expressed in the form of (2) can be determined by the Lie Algebra Rank Condition [3]. From another point of view a *nonholonomic* system is obeying one or more nonintegrable equality constraints above the derivatives of the configuration variables. This intuitively means that the configuration velocity vector is locally constrained to some directions while it is not possible to directly move in other directions (think for example of driving a car sideways).

An interesting property of nonholonomic systems is that they are underactuated but remain still controllable on the whole state space. Local control methods which determine the control signal $u(\cdot)$ for a given (q_I, q_G) pair in the absence of obstacles are called *steering methods*.

2.2 Steering Methods

The steering of nonholonomic systems is a difficult task even in the absence of obstacles. Exact algorithms are not available in general but only for special classes of systems, e.g. nilpotentizable [5], chain-formed [6–8] and differentially flat systems [9]. Common examples for the above mentioned systems include wheeled vehicles with or without trailers.

An important family of steering approaches is based on optimal control. For general systems only approximate methods exist [10]. Exact methods for computing optimal (e.g. shortest length) local paths are available only for car-like robots moving forward (Dubins-car [11]) or both forward and backward (Reeds–Shepp-car [12–15]) and for robots equipped with differential drive [16, 17]. For the Reeds-Shepp-car, it is shown in [12] that for any pair of configurations the shortest path can be chosen from 48 possible sets of paths, each of which consisting of maximum five circular or straight segments and having maximum two cusps. The number of sets were reduced to 46 in [13] and to 26 in [14].

2.3 Nonholonomic Planning among Obstacles

To design a planning algorithm that deals with both kinematic and environmental constraints, global information about the connectivity of the free space has to be taken into account, together with a well-suited steering method at the same time. Numerous exact and sampling-based, probabilistic and deterministic global planning methods have been developed for systems without nonholonomic constraints [1, 3], whose ideas can be adapted to the nonholonomic case. Motion planning for nonholonomic systems induces two main questions. The first question addresses the *existence* of a collision-free and feasible path, while the other problem to solve is the *synthesis* of such a path. It is shown in [2, 4] that the existence of an admissible path between two configurations is equivalent to the existence of any collision-free path between the same configurations.

To the best knowledge of the authors, there is no *optimal* solution available for the synthesis problem. The majority of algorithms delivering a *feasible* solution can be grouped in two main categories. Sampling-based roadmap methods [18, 19] belong to the first one, which build a graph in order to capture the topology of the free space and use local steering methods to connect the graph nodes. The second category consists of techniques that exploit the idea of the above mentioned equivalence between the existence of pure geometric and nonholonomic paths. These methods [20–22] approximate a not necessarily admissible but collision-free initial path by a sequence of feasible paths.

2.3.1 Approximation Methods

In the framework proposed in [20] the initial path is recursively subdivided until every part can be replaced by a collision-free admissible path obtained by an LPM. This work deals with car-like robots and uses an LPM based on the shortest paths for the Reeds–Shepp-car. It is shown in [20] that the number of subdivisions is inversely proportional to the square of the clearance of the initial path. According to this, paths going far from obstacles are well-suited to approximation methods.

This property is expolited in [21], where firstly a maximum-clearance skeleton of the free space is obtained and used for initial path planning. Secondly, the sequence of admissible sub-paths is chosen based on maximal collision-free balls defined by the Reeds–Shepp metric.

The quality of the resulting approximated feasible paths is affected by the "goodness" of the initial geometric path. If the initial path leads mostly in directions which are non-admissible for the system, then the approximated path will be quite difficult (containing a great amount of reversals). Therefore usually a final optimization of the approximated path is necessary (e.g. in [20] pairs of intermediate configurations are randomly chosen and tried to be connected by simpler local paths). An approach for increasing the "goodness" of the initial path based on nonholonomic deformation of potential fields can be found in [22].

The above mentioned approximation methods are complete which means that they deliver a solution if one exists and report failure if not. The only precondition to achieve this is the existence of an unconstrained path with nonzero clearance about it. The drawbacks are high computational cost and complexity of the resulting path.

2.3.2 Other Approaches

Some improvements can be achieved in terms of path complexity and computation time if the need for completeness is not essential. Two recently proposed approaches are mentioned here which were inspiring in the development of our algorithm. In [23] a practical method is presented for planning parking maneuvers of car-like robots in narrow environments, based on the idea of reducing the free space to regions that are reachable by a concatenation of simple motions. One motion primitive is a straight–circular–straight triplet, and the resulting solution is chosen from a set obtained by a breadth-first search of candidate sequences of these primitives. The choice is the result of a numerical optimization of an objective function consiting of more weighted terms. This method is practical in situations if two or three consecutive motion primitives suffice to reach the goal (e.g. parallel or perpendicular parking tasks) but the computational cost increases strongly with the number of required primitives (denoted as "depth" of the path in [23]).

The approach presented in [24] works well in cluttered but not very narrow environments. It employs cell decomposition algorithms to abstract the environment to a topological roadmap [25] which is used to obtain a preliminary path. This path consists of straight segments and is made feasible for car-like robots by simply smoothing the corners by circular arcs of minimum admissible turning radius. This implies that the resulting path contains no cusps. If a path cannot be smoothened this way, or the resulting path intersects with obstacles, the path is simply rejected and another one is searched in the roadmap. The algorithm proceeds until a feasible solution is found, or it reports failure if it runs out of candidate preliminary paths.

Our proposed approach can be treated as an approximation method. It borrows ideas from both [23] and [24]. A preliminary path is obtained by cell decomposition, which serves as a guide for the local planner module. The LPM uses a reduced set of path primitives, consisting of two or three circular and straight segments. This affects the completeness of the method but benefits from the reduced computational burden. The computational cost does not increase dramatically with the number of concatenated local paths thus difficult situations in narrow environments, requiring more maneuvers can be solved.

3 Problem Statement

The aim of our paper is to present a path planning algorithm for car-like robots in situations where narrow corridors have to be passed.

The kinematic model of a car-like robot is depicted in Figure 1. Its instantaneous configuration can be completely described by $q = [x,\ y,\ \theta]^T$ in $C = \mathbb{R}^2 \times \mathbb{S}^1$. The equations of motion are the following:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\phi \end{aligned} \tag{3}$$

where v is the signed velocity of the back axle midpoint (positive means forward motion), ϕ denotes the steering angle and L the distance of the front and rear wheel axles. The absolute value of the steering angle is limited to $|\phi| \le \phi_{max} < \pi/2$.

If we perform an input transformation and define $u_1 = v$ and $u_2 = \frac{v}{L}\tan\phi$, the same form as (2) is obtained with the following substitutions: $x = q$ (the state is the configuration itself), $h_1(q) = [\cos\theta,\ \sin\theta,\ 0]^T$, $h_2 = [0,\ 0,\ 1]^T$. The control inputs u_1 and u_2 correspond to the translational and angular velocities of the center of the rear axle, respectively. This point is considered as the reference point of the car.

The radius ρ of the circular path traversed by the reference point if the steering angle is fixed can be determined from basic

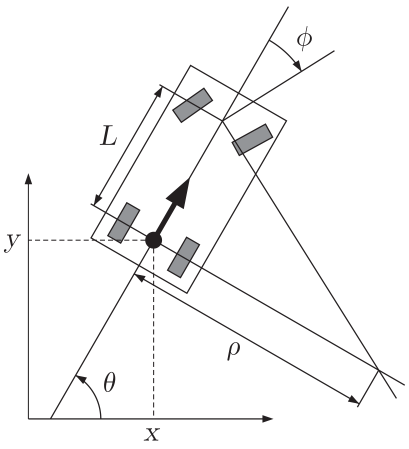


Fig. 1. Model of a car-like robot

trigonometry:

$$\rho = \frac{L}{\tan\phi} \tag{4}$$

Note that ρ can be positive or negative according to the sign of the steering angle ϕ. In the depicted situation $\phi < 0$ thus $\rho < 0$ as well (intuitively this means "turning right"). As the steering angle tends to zero, the turning radius tends to infinity, which corresponds to straight motion. The constraint on the steering angle defines a minimum turning radius constraint:

$$|\rho| \geq \rho_{min} = \frac{L}{\tan\phi_{max}} \tag{5}$$

It should be noted that in the current presentation we cope only with the kinematic properties of the robot, which is represented by the minimum turning radius constraint. The geometric shape of the vehicle is neglected and path planning for the reference point is performed. Although this is not practical, it does not affect the validity of the underlying idea of the presented algorithm. An extension to polygonal robot shape causes no theoretical complications (the collision checking algorithm presented in Section 6.1 has to be adapted to this case).

Our goal is to obtain feasible paths being possibly simple, i.e. containing a low number of maneuvers. Narrow environments and simple paths are conflicting concepts, it is enough to think of turning around with a car in a narrow street. The "narrowness" of the free space is a somewhat qualitative concept thus it is worthwhile to give a more quantitative definition to it.

Definition 1 (Local Narrowness)**.** *The* local narrowness *is a positive real-valued function η over the configuration space, which assigns to every configuration q the value*

$$\eta(q) = \begin{cases} \dfrac{2\rho_{min}}{R_{max}(p(q))}, & \text{if } q \in C \setminus C_{obs}, \\ \infty & \text{otherwise}, \end{cases} \tag{6}$$

where $R_{max}(p(q))$ is the radius of the maximal collision-free disk in $\mathbb{R}^2$, centered at the position $p(q) = (x, y)$ belonging to q.

An illustration of this function is shown in Figure 2. The dark gray area corresponds to local narrowness values $\eta \leq 1$, at these configurations we say that the environment is "wide". This intuitively means that starting from this area – without respecting orientation – the robot can travel along a whole circle of radius ρ_{min} witout collision. Function values between 1 and 2 are depicted by light gray and stand for "narrow" places. Starting from these areas the robot is assured to have a collision-free angular displacement of at least $\pi/2$ along a minimum-radius circular arc at least in one direction of the four possibilities (forward–left, forward–right, backward–left or backward–rigt). Intuitively, when traveling in a corridor with parallel walls, if the robot is at a "narrow" position, it has the possibility to turn back with no more than one reversal. In the white area the local narrowness is $\eta \geq 2$ denoted by "very narrow", which means that the size of the free space around the robot (represented by R_{max}) is not greater than the minimal turning radius. If going to these locations, the chance of simple maneuvering decreases.

Definition 2 (Global Narrowness)**.** *The* global narrowness *of a planning query* $\Gamma(C, q_I, q_G, \rho_{min})$ *is defined by*

$$\eta(\Gamma) = \min_{\lambda\in\Lambda}\left(\max_{s\in[0,1]}\bigl(\eta(\lambda(s))\bigr)\right). \tag{7}$$

This means that if an environment is given – represented by C – together with a query pair (q_I, q_G) and a minimal turning radius ρ_{min} (which are called together as a *planning query* Γ), no smaller maximal local narrowness can be achieved along any geometric path $\lambda \in \Lambda$ than $\eta(\Gamma)$. The global narrowness can be treated as a measure of the difficulty of a planning query.

Our aim is to obtain a planning algorithm that succesfully solves planning queries with $\eta(\Gamma) \gg 2$ for point-shaped car-like robots, especially in environments where $\eta(q) \geq 1$ for all $q \in C$. These problems can be treated as "difficult" because the robot has to pass "very narrow" places (e.g. corridors of small width), while the wider areas between them are still "narrow".

4 Local Feasible Planning

The most widely used local planners for car-like robots are based on the finite set of shortest-length paths for the Reeds–

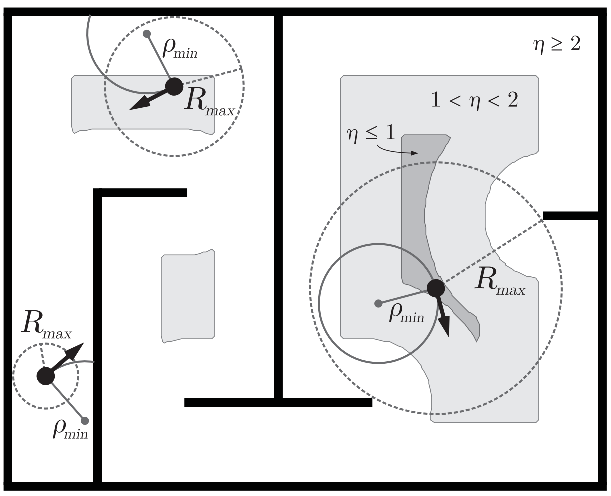


Fig. 2. The concept of local narrowness

Shepp-car [12, 15]. These paths consist of maximum five circular and straight segments, where the radius of circular arcs is exactly ρ_{min} and there are at most two cusps. The optimal steering method based on these paths verifies the *topological property* [2] which means that as q_I and q_G get closer, the radius of the minimal disk that contains the resulting path gets smaller. This property ensures the possibility to construct complete approximation methods based on this steering method, such as [20]. Using the set of Reeds–Shepp paths results in an optimal-length solution in the absence of obstacles, but if obstacles are present, it becomes necessary to obtain a concatenation of these. This is possible – according to the topological property – even in case of narrow planning queries, however, the concatenated path can consist of a high number of parts with unnecessarily many reversals, resulting in a solution being complex and far from optimal. According to this, it seems appealing to investigate other possibilities beyond the set of optimal local paths.

Let us denote a path consisting of circular arcs and straight segments by a sequence of S (for straight) and C (for circular) letters. For example, $SCCS$ means a path consisting of a starting straight segment, two consecutive circular arcs and a final straight segment. The following lemma holds:

Lemma 1. *In the absence of obstacles, any two configurations q_I and q_G of a car-like robot can be connected by a feasible path consisting of no more than three segments. The solution can be found in the form of an SCS or CCS path.*

Proof. Let us denote the position of a configuration q by $p(q) = [x(q), y(q)]^T$ and the orientation of the same as $\theta(q)$. The main idea of our local planning approach is that if q_I is not on the line going through $p(q_G)$ and having the orientation $\theta(q_G)$ (let us denote this *the line of* q_G), then it is possible to reach an intermediate configuration q'_G by one circular path segment, which belongs to the same line. As can be seen in Figure 3, there are two such configurations $q'_{G,1}$ and $q'_{G,2}$, reachable through four possible arcs. These arcs can be obtained by drawing circles tangential to q_I and the line of q_G as well. One of the intermediate configurations has opposite orientation than q_G, this is called *invalid* and can be discarded. From the *valid* intermediate configuration ($q'_{G,1}$ in Figure 3) q_G can be trivially reached through a straight path segment. The resulting path is CS (which can be treated as SCS or CCS as well, with a zero-length first segment).

Until now, the minimal turning radius has not been taken into account. If $p(q_I)$ is not on the line of q_G and the circular arc leading to the valid q'_G has a radius $|\rho| < \rho_{min}$, then another intermediate configuration q'_I can be obtained in advance by an S or C path starting from q_I. This new configuration q'_I can be chosen such that a next valid intermediate configuration q'_G can be reached by a circular arc of $|\rho| \geq \rho_{min}$ as depicted in Figure 3. The resulting path from q_I to q_G will thus have the form of SCS or CCS.

If $p(q_I)$ is on the line of q_G, there are two possibilities. In the case when $\theta(q_I) = \theta(q_G)$, the solution path consists of only one straight segment (that is identical with SCS or CCS having two zero-length segments). In the other case, when $\theta(q_I) \neq \theta(q_G)$, an appropriate intermediate configuration q'_I can be obtained by an S or C segment as described above. The result is again an SCS or CCS path. □

The reason why local paths ending with a straight segment are used will be clear in the next two sections. It is enough to know at the moment that the LPM is used together with a global planner that obtains a preliminary path consisting of straight segments. Using SCS or CCS local paths, it is possible to arrive in every step of the planning process on a following straight segment.

To construct an SCS or CCS path one has to accomplish three steps according to the three path segments. The final step of constructing the last S segment is trivial if the preceding two are available. The second path segment (which is always C) can be easily constructed by simple geometrical operations as depicted in Figure 3. The construction of the first S or C path segment is described in the sequel.

The set of configurations reachable from a given configuration q_0 by one circular arc or straight segment is the *Arc Reachable Manifold* of q_0 (ARM_{q_0}), as defined in [26]. This is a two-dimensional submanifold of the $\mathbb{R}^2 \times \mathbb{S}^1$ configuration space. If $q_0 = [0, 0, 0]^T$ and a position $p = [x, y]^T$ is given, the orientation of the configuration at p can be determined by

$$\theta(p) = \begin{cases} \operatorname{atan2}\left(x, \frac{x^2-y^2}{2y}\right), & \text{if } y > 0, \\ -\operatorname{atan2}\left(x, -\frac{x^2-y^2}{2y}\right), & \text{if } y < 0, \\ 0, & \text{if } y = 0. \end{cases} \tag{8}$$

The radius of the path leading from q_0 to p is

$$\rho(p) = \frac{x^2 + y^2}{2y}. \tag{9}$$

If $q_0 \neq [0, 0, 0]^T$, the same calculations can be done after a coordinate transformation to the local frame of q_0.

As can be seen, if no obstacles are present, for a given initial configuration q_I any position in the workspace $W \subset \mathbb{R}^2$ can be reached by a circular arc ($|\rho| < \infty$) or straight segment ($|\rho| = \infty$).

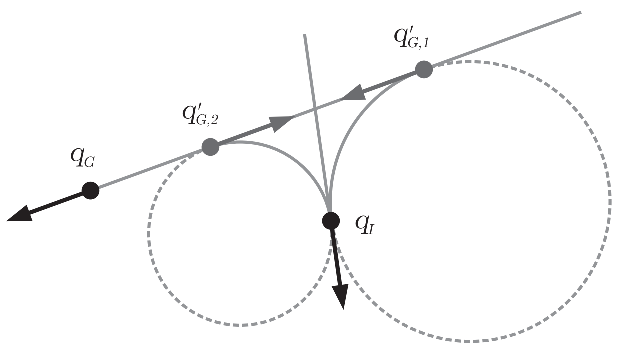


Fig. 3. Local CS path between two configurations

Hence there are uncountably infinite possible solutions for the first segment of an SCS or CCS path of our LPM, even if we discard the inadmissible elements of ARM_{q_I} that have $|\rho| < \rho_{min}$.

For this reason the LPM performs a numerical search to obtain the local path. On one hand a solution has to be found that does not intersect with obstacles, on the other hand the result should be optimal in a sense as well. This optimality criterion is the path length in our case (note that the obtained local path will not be optimal in the sense of Reeds and Shepp but only among SCS/CCS paths). The operations of the LPM are summarized in Algorithm 1.

Algorithm 1 LPM for SCS/CCS paths

```
1: Discretize the workspace of the robot uniformly, let the set
   of discrete positions be $P$
2: for all positions $p_i \in P$, $i = 1 \dots K$ do
3:   Determine the circular or straight path segment $\lambda_{1,i}(q_I, p_i)$
     leading from $q_I$ to $p_i$
4:   if $|\rho(p_i)| \geq \rho_{min}$ AND $\lambda_{1,i}(q_I, p_i) \cap C_{obs} = \emptyset$ then
5:     Determine the four possible circular path segments
       $\lambda_j(q'_I, q'_{G,k})$, $j, k = \{1, 2\}$ tangential to $q'_I = \left[p_i^T, \theta(p_i)\right]^T$
       and the line of $q_G$
6:     for all path segments $\lambda_j(q'_I, q'_{G,k})$ do
7:       if $\left|\rho\left(\lambda_j(q'_I, q'_{G,k})\right)\right| \geq \rho_{min}$ AND $\lambda_j(q'_I, q'_{G,k}) \cap C_{obs} =$
         $\emptyset$ then
8:         Set $d_{j,k}$ to the length of $\lambda_j(q'_I, q'_{G,k})$
9:       else
10:        Set $d_{j,k} = \infty$
11:      end if
12:    end for
13:    Let $d_{2,i} = \min\left(d_{j,k}\right)$
14:    Let $\lambda_{2,i} = \lambda_j(q'_I, q'_{G,k})$ belonging to $d_{2,i}$
15:    Let $q'_G = q'_{G,k}$ belonging to $\lambda_{2,i}$
16:    Determine the straight path segment $\lambda_{3,i}(q'_G, q_G)$
17:    if $\lambda_{3,i}(q'_G, q_G) \cap C_{obs} = \emptyset$ then
18:      Set $d_{3,i}$ to the length of $\lambda_{3,i}(q'_G, q_G)$
19:    else
20:      Set $d_{3,i} = \infty$
21:    end if
22:    Set $d_{1,i}$ to the length of $\lambda_{1,i}(q_I, p_i)$
23:  else
24:    Set $d_{1,i} = \infty$
25:  end if
26: end for
27: Let $d = \min_{p_i \in P} (d_{1,i} + d_{2,i} + d_{3,i})$
28: if $d = \infty$ then
29:   return FAILURE
30: else
31:   Let $p^* = \arg\min_{p_i \in P} (d_{1,i} + d_{2,i} + d_{3,i})$
32:   Let $\lambda_{local}(q_I, q'_I, q'_G, q_G)$ be the concatenated path be-
      longing to $p^*$
33:   return $\lambda_{local}(q_I, q'_I, q'_G, q_G)$
34: end if
```

This algorithm takes K samples from the workspace uniformly and determines the intermediate configuration q'_I at each sampling point p, according to (8) and (9) (this will be the final configuration of the first S or C segment). From each intermediate configuration the four CS paths to q_G are computed as depicted in Figure 3. The resulting local path is the shortest collision-free one chosen from the $4K$ candidate SCS/CCS paths. In our current implementation we do not make any distinction between valid and invalid intermediate goal configurations q'_G for simplicity. This causes the resulting path arrive at the goal configuration q_G or at a modified goal configuration $\hat{q}_G$ having the same position but opposite orientation. This limitation will be eliminated in a later version of the planning algorithm.

5 Preliminary Global Path

Planning a preliminary global path without taking nonholonomic constraints into account is a key component of approximation-based nonholonomic planners. A lot of geometric planning approaches are available in the literature, a comprehensive survey of these can be found in [1] or [3]. There exist sampling-based approaches and exact methods as well, each having different advantages and limitations. Exact methods are usually effective in case of low-dimensional configuration spaces (e.g. a rigid robot moving in the plane). In case of higher dimensions (e.g. planning for robots consisting of more links) they become computationally intractable and sampling-based methods perform better. We summarize some exact approaches for two-dimensional problems ($C = \mathbb{R}^2$) in the sequel.

Perhaps the oldest methods are *shortest-path roadmaps* based on mutual visibility of candidate pathpoints. The *visibility graph* method or its advanced *tangent graph* version [27] builds a topological graph whose nodes are obstacle corners and edges are collision-free straight segments between them. The initial and goal points are connected to the graph in this way as well. A shortest path search in this graph results an optimal-length path. Unfortunately this implies that the path touches obstacle corners, which is not an appealing property in practical applications.

An alternative to shortest-path methods are *maximum clearance roadmap* methods. These try to keep the path as far as possible from obstacles. An example is the *generalized Voronoi diagram* or *retraction method* [28]. This results in a safe path consisting of straight segments and parabolic curves.

Cell-decomposition methods are partitioning the free configuration space into a finite set of regions called *cells*. This yields again a graph-based abstraction of the environment. The regions correspond to graph vertices and edges represent the adjacency of these. A decomposition is useful if a path between two points inside the same cell can be trivially obtained (e.g. if the cells are convex then straight path segments suffice). There

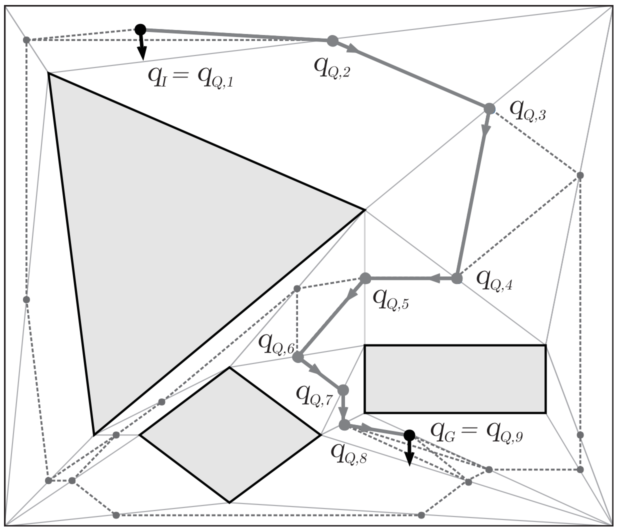


Fig. 4. Preliminary path based on triangular cell decomposition

are many such methods available differing in the shape of cells (e.g. trapezoidal, triangular, polytopal decompositions). A survey and comparison of these can be found in [24, 25].

Algorithm 2 describes our global planner module which assumes polygonal environment and performs a constrained Delaunay triangulation of the free space. We use the freely downloadable Delaunay triangulator program *Triangle* [29, 30] as a subroutine in our algorithm. This returns the set of triangles $T = \{t_1 \dots t_n\}$ and an $n \times n$ adjacency matrix A belonging to them. Based on the triangular decomposition a roadmap (topological graph) $G = (V, E)$ is constructed. The set of graph vertices V contains the median points of the adjacent triangle edges and the initial and goal positions. The set E of graph edges consists of straight segments connecting vertices belonging to the same triangle. The initial and goal positions are connected to the graph vertices belonging to their containing triangle. Edges are weighted by their Euclidean length. Finally, a shortest path search is performed using Dijkstra's algorithm [31]. The shortest path is represented by a $3 \times m$ matrix Q, where every column corresponds to a configuration $q_{Q,k} = [x_Q, k, y_{Q,k}, \theta_{Q,k}]^T$, $k \in \{1 \dots m\}$ representing corner points. The orientation (third element of $q_{Q,k}$) of corner points between $q_{Q,1} = q_I$ and $q_{Q,m} = q_G$ is determined by the angle of the path segment following them. An example can be seen in Figure 4. The triangle boundaries are drawn by thin solid lines, the dashed line shows the roadmap and the shortest path in the roadmap is depicted by a strong solid line.

Algorithm 2 Preliminary path planning

```
 1: Initialize $V = \emptyset$ and $E = \emptyset$
 2: $\{T, A\} = \text{Triangle}(C \setminus C_{obs})$, $n = |T|$
 3: for all $i \in \{1 \dots n\}$, $j \in \{i + 1 \dots n\}$ do
 4:     if $A_{i,j} = 1$ then
 5:         Insert the median point of the common edge of $t_i \in T$
            and $t_j \in T$ in $V$
 6:     end if
 7: end for
 8: for all $i \in \{1 \dots n\}$ do
 9:     Determine the set of graph vertices $V_{t_i} \subset V$ which belong
        to $t_i$
10:     for all $k \in \left\{1 \dots \left|V_{t_i}\right|\right\}$, $l \in \left\{i + 1 \dots \left|V_{t_i}\right|\right\}$ do
11:         Insert the edge connecting $v_k \in V_{t_i}$ and $v_l \in V_{t_i}$ in $E$
12:     end for
13: end for
14: Insert $v(q_I) = p(q_I)$ and $v(q_G) = p(q_G)$ in $V$
15: Determine the triangles $t_I$ and $t_G$ which contain $p(q_I)$ and
    $p(q_G)$, respectively
16: Insert edges in $E$ that connect $v(q_I)$ and $v(q_G)$ to the vertices
    belonging to $V_{t_I}$ and $V_{t_G}$, respectively
17: $Q = \text{ShortestPath}(G(V, E),\ v(q_I),\ v(q_G))$
18: for all $k \in \{2 \dots m - 1\}$ do
19:     $\theta_{Q,k} = \text{atan2}((y_{Q,k+1} - y_{Q,k}),\ (x_{Q,k+1} - x_{Q,k}))$
20: end for
21: return $Q$
```

The resulting path is used for guiding the local planner described in Section 4. It is appropriate for this purpose because it goes far enough from obstacles (although it is not optimal with respect to obstacle distance). This property is ensured by the path construction method. Since the triangles are spanned by corner points of obstacles, the path consists of line segments connecting midpoints between obstacle corners and being completely in the interior of the free space.

6 Overview of the Planning Algorithm

The point of our nonholonomic planning algorithm is the iterative execution of the local planner module, guided by the preliminary global path. To avoid confusion in the notation, let us use q_I and q_G for the initial and goal configurations of the global planning query. Let us denote the configurations q_I, q_I', q_G' and q_G of a local path by q_{IL}, q_{IL}', q_{GL}' and q_{GL}, respectively.

An important property of the planning algorithm is that it does not force the final path to go through intermediate corner points of the preliminary path. This is achieved by a slight modification of the LPM described in Section 4. Instead using local SCS/CCS paths between intermediate points, the last straight segment is omitted and the next local path is started at the intermediate configuration q_{GL}' of the current local path. We do this because it is not important to traverse the intermediate corner points of the preliminary path at all, just to travel along the lines spanned by the path segments. The only exception to this is the case when the local goal configuration q_{GL} is equal to the global goal q_G. Thus in fact the resulting feasible global path consists of several SC/CC paths and one final SCS/CCS path. For this reason we denote the local paths by $SC(S)/CC(S)$ in the sequel to emphasize that in most cases the last straight segment has not to be added to the final path. Note that the local planner module always computes the whole SCS/CCS path (collision detection and length optimization occurs for the three-segment version)

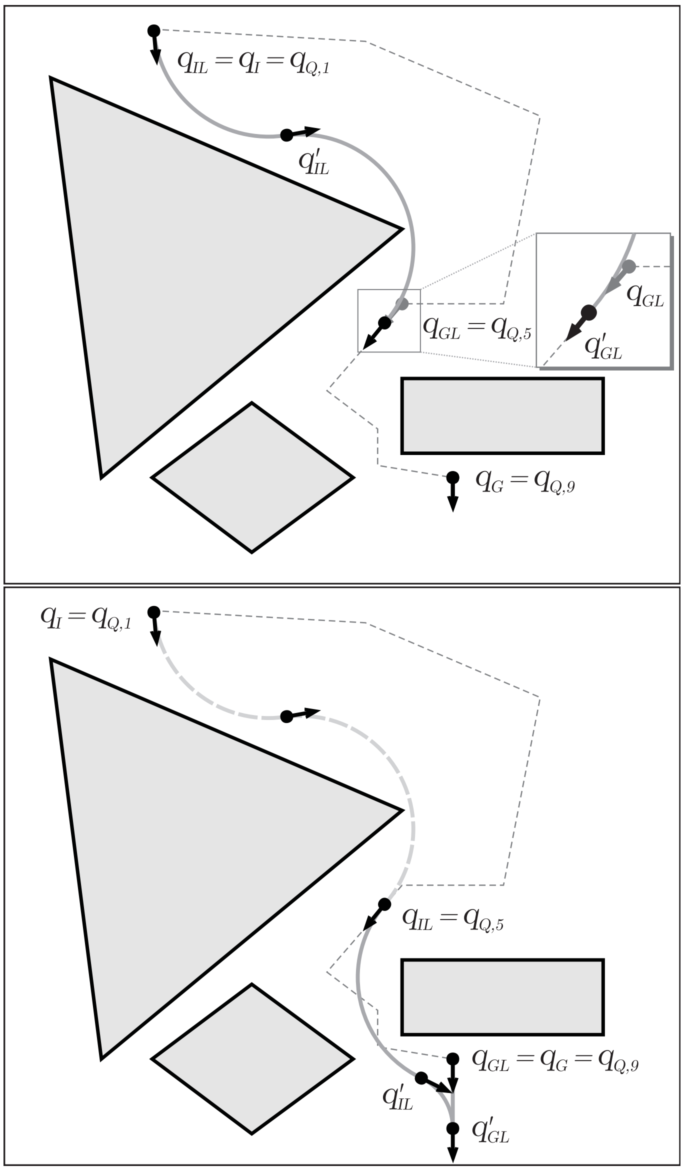


Fig. 5. Execution of the planning algorithm (a) Step 1: $q_{IL} = q_I$, $q_{GL} = q_{Q,5}$,(b) Step 2: $q_{IL} = q_{Q,5}$ (updated), $q_{GL} = q_G$

and the last S segment is cut away afterwards.

6.1 The Planning Process

The planning process is described by Algorithm 3 and an example is shown in Figure 5 for the situation already seen in Figure 4. The final path λ_{final} is represented by a $6 \times \ell$ matrix, where each column represents a single S or C path segment $\lambda_k = \left[q_k^T, \rho_k, \Delta\theta_k, d_k\right]^T$, $k \in \{1 \ldots \ell\}$, where q_k is the starting configuration of λ_k, ρ_k is the radius ($\rho_k = \infty$ in case of S), $\Delta\theta_k$ is the change in the orientation and d_k is the travel distance along the path segment. The latter is given by

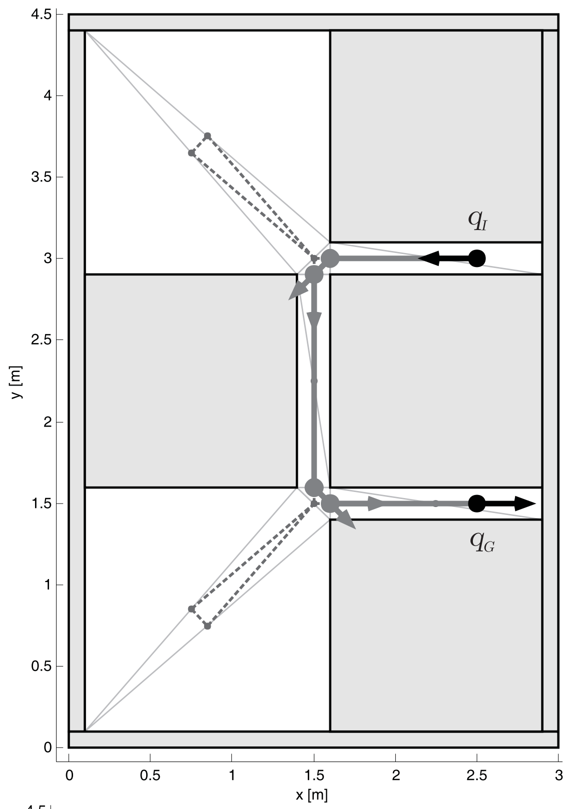
$$d_k = \begin{cases} \rho_k \Delta\theta_k, & \text{if } \lambda_k \text{ is } C, \\ \left|p(q_{k+1}) - p(q_k)\right|, & \text{if } \lambda_k \text{ is } S, \end{cases} \tag{10}$$

A local path λ_{local} is represented by a 6×3 matrix with the same meaning of columns. The local path without its last S segment (without the last column in the matrix representation) is denoted by λ^*_{local}. If a new local $SC(S)/CC(S)$ path is found during the execution of the algorithm, it is simply appended to the existing path by adding the columns of λ^*_{local} or λ_{local} to λ_{final}. If a local path is found that ends at some intermediate configuration $q_{Q,j}$ of the preliminary path, then $q_{Q,j}$ is replaced by q'_{GL} (the final configuration of λ^*_{local}) and the next local planning step starts here (see line 11 of Algorithm 3). It can be seen in the example in Figure 5a. that the last S segment of the local path would cause an unnecessary reverse motion hence it makes sense to continue the planning process from q'_{GL} instead of q_{GL}.

The planning algorithm strives to found a path consisting of the least possible number of segments. For this reason always the global goal configuration q_G is chosen first as local goal for the LPM. If a solution is found, the planning process terminates and returns the final path. If there is no solution, then a new local goal is chosen from the preliminary path by halving the number of remaining segments (line 18 of Algorithm 3). In the example depicted in Figure 5 the final path was obtained by two consecutive local paths. Since q_G was not reachable from q_I (the LPM reported failure), $q_{Q,5}$ was chosen as the next local goal because there was 8 remaining segments in the preliminary path and $q_{Q,5}$ was at the end of the fourth segment. Since a $CC(S)$ solution was found to $q_{Q,5}$ (Figure 5a), it was updated to $q_{Q,5} = q'_{GL}$ and the local planning was initiated from here with q_G as local goal again (Figure 5b). Though the planning process was guided by the preliminary path, the resulting final path has only one common configuration with it ($q_{Q,5}$), except q_I and q_G.

Algorithm 3 Final feasible path planning

```
 1: $Q = [q_{Q,1} \ldots q_{Q,m}]$ = PreliminaryPath $(C_{obs}, q_I, q_G)$
 2: Initialize $\lambda_{final} = \emptyset$, $i = 1$, $j = m$
 3: loop
 4:   Set $q_{IL} = q_{Q,i}$ and $q_{GL} = q_{Q,j}$
 5:   if $\lambda_{local}$ = LPM $(C_{obs}, q_{IL}, q_{GL})$ exists then
 6:     if $q_{GL} = q_G$ then
 7:       $\lambda_{final}$ = append $(\lambda_{final}, \lambda_{local})$
 8:       return $\lambda_{final}$
 9:     else
10:       $\lambda_{final}$ = append $(\lambda_{final}, \lambda^*_{local})$
11:       Set $q_{Q,j} = q'_{GL}$
12:       Set $i = j$ and $j = m$
13:     end if
14:   else
15:     if $j - i = 1$ then
16:       return FAILURE
17:     else
18:       $j = \lceil \frac{j-i}{2} \rceil + i$
19:     end if
20:   end if
21: end loop
```



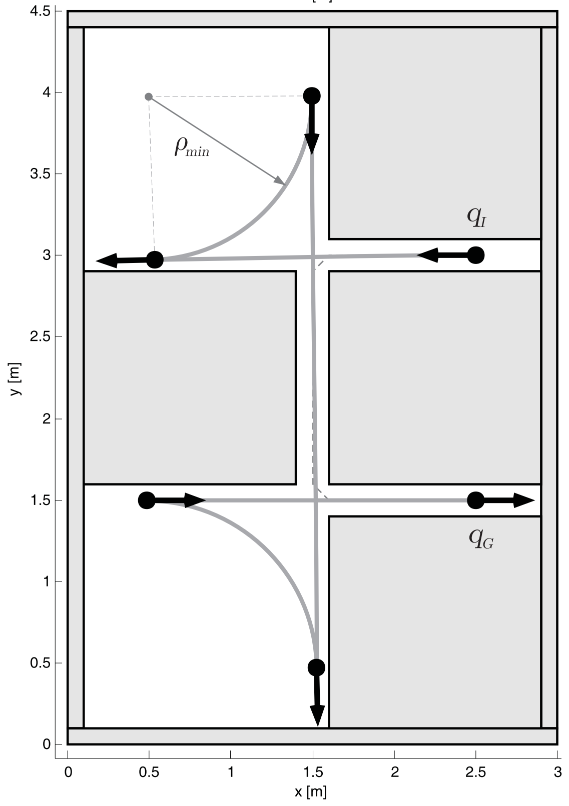


Fig. 6. Test scenario 1: $\min(\eta(q)) = 2.667$, $\eta(\Gamma) = 20$ (a) Cell decomposition, roadmap and preliminary path, (b) Resulting path

6.2 Simulation Results

The proposed algorithm was tested in simulations for environments containing narrow corridors. It proved to be a reliable and useful algorithm in the sense that it produced reasonable paths, similar to the ones a human driver would obtain.

Figure 6 shows a scenario where three narrow lanes have to be traversed to reach the goal. The wider regions in the upper and lower left corners can be used for maneuvering. The width of these is $1.5m$ thus the maximum value for R_{max} in these areas is $0.75m$. The width of the corridors is $0.2m$ thus $R_{max} \leq 0.1m$ inside them. The minimum turning radius is set to $\rho_{min} = 1m$ hence the local narrowness (6) is not less than $2/0.75 = 2.667$ for all $q \in C$, which means that the environment is "very narrow" everywhere. The global narrowness of the planning query (7) is $\eta(\Gamma) = 20$ according to the narrow corridors. The resulting path strongly deviates from the preliminary path in the wider areas, which is a reasonable and necessary choice that ensures a successful arrival at the goal.

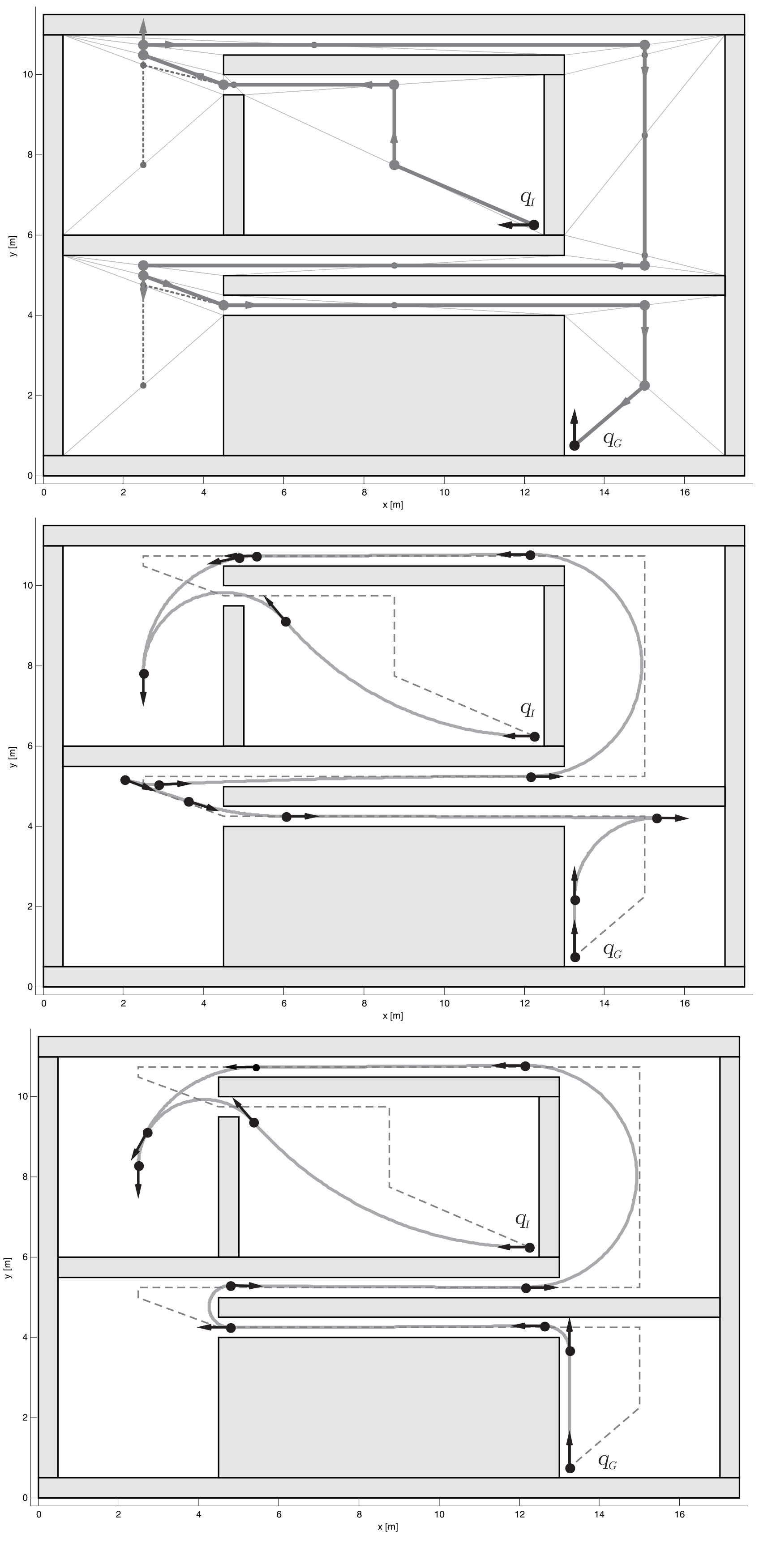
Another test scenario is depicted in Figure 7. In this case a more complex environment was chosen with several bottlenecks and wider regions. The maximum of R_{max} is $2m$ in the wider areas, and $R_{max} \leq 0.25m$ in the corridors. Two results are presented, for different turning radius limits. In Figure 7a $\rho_{min} = 2m$ which results in a local narrowness $\eta(q) \geq 2$ and a global narrowness $\eta(\Gamma) = 8$. Hence this environment and planning query can be designated as “very narrow”, similar to the previous example. The preliminary path consists of 13 segments and the planner found a reasonable solution consisting of 6 local $SC(S)/CC(S)$ paths. Perhaps the only difference between this and a solution obtained by a fictitious human driver would be that he would not prefer backward motion in long corridors. Nevertheless, this is a normal behavior since backward motion is not penalized in the algorithm. Another case with a relatively small ρ_{min} is depicted in Figure 7c. The turning radius is allowed to be as little as $0.5m$ which enables direct U-shaped turns at the end of the corridors. As can be seen, the algorithm exploited this opportunity and returned a path that contains some of these sharp turns.

6.3 Complexity and Completeness

The computational complexity of the proposed algorithm is determined by different things. The algorithm can be divided in two main phases: the preliminary planning and the approximation (iterative local planning) parts. The complexity of both is examined in the sequel.

The Delaunay triangulator algorithm has a complexity of $O(N \log N)$, where N is the number of obstacle vertices [29]. It can be easily derived from Euler’s polyhedron formula that the number of triangles in any triangulation of N points is $M = 2N - 2 - b$, where b is the number of points being on the convex hull of the point set. This value is an upper bound in case of constrained Delaunay triangulation. If the environment is rectangle-shaped ($b = 4$) then $M \leq 2N - 6$. It can be derived similarly that in case of $b = 4$ the number of triangle edges is $S \leq 3N - 7$. The roadmap for the preliminary path planner is constructed of the midpoints of the triangle edges and the initial and goal positions thus the number of roadmap vertices is $|V| = S + 2$. Since a subset of roadmap edges are connecting vertices belonging to the same triangle and every triangle has

Fig. 7. Test scenario 2. (a) Cell decomposition, roadmap and preliminary path (b) Resulting path in case of $\rho_{min} = 2m$, $\min(\eta(q)) = 2$, $\eta(\Gamma) = 8$, (c) Resulting path in case of $\rho_{min} = 0.5m$, $\min(\eta(q)) = 0.5$, $\eta(\Gamma) = 2$



maximum three vertices, the number of these roadmap edges cannot be greater than $3M$. There are at least four "outer" triangles at the boundary of the environment which have maximum two roadmap vertices and thus only one roadmap edge. For this reason the maximum number of edges can be lowered to $3M - 8$. The initial and goal vertices are connected to the vertices belonging to their containing triangles, which adds maximum 6 new edges to the previous amount. Hence and upper bound to the number of roadmap edges is $|E| \leq 3M - 2$. The best known worst-case complexity of Dijkstra's shortest path search algorithm is $O(|E| + |V| \log |V|)$ [32], which is $O(N \log N)$ because both $|E|$ and $|V|$ are $O(N)$ according to the above calculations. Thus the preliminary global planning algorithm is $O(N \log N)$.

The local planner module samples the free configuration space uniformly and calculates a local path for each sampling point (where the sampling point itself is the position of the intermediate configuration q'_I). A local path through a given sampling point can be computed in constant time and the collision check requires $3N$ time because intersections of three local path segments with N obstacle boundary segments has to be checked. If we use a grid-based sampling with k_x and k_y samples along the x and y axes, then the complexity of the LPM becomes $O(k_x k_y N)$. The number of edges in the preliminary path is upper bounded by the number of vertices in the roadmap which is $O(N)$. For one local initial configuration q_{IL} the local goals of the LPM are selected by halving the number of remaining preliminary path segments. This causes an $O(\log N)$ upper bound on the number of trials until a local path is found or failure is reported. Since the number of preliminary path edges is $O(N)$, the worst-case number of LPM exectuions is $O(N \log N)$. This yields an $O(k_x k_y N^2 \log N)$ overall worst-case complexity of the approximation phase. Since this outweighs the preliminary planning phase, it can be stated that the worst-case complexity of the whole planning algorithm is $O(k_x k_y N^2 \log N)$.

The algorithm does not verify the completeness property. Situations can be constructed where it fails to obtain a solution although it can be shown that one exists. Most of these situations are related to ill-orientated initial or goal configurations. One such situation is depicted in Figure 8. If the initial configuration of the LPM is in the middle of a narrow corridor whose width is smaller than ρ_{min} (which means $R_{max} < \rho_{min}/2$ and $\eta(q_{IL}) > 4$) and the line of the next local goal configuration is parallel with the corridor walls but the initial orientation is perpendicular to them, then no $SC(S)/CC(S)$ path exists that would lead to q_{GL}. In Figure 8 the boundary case of $R_{max} = \rho_{min}/2$ is depicted which is the narrowest case for which a solution can be found with such initial and goal configurations.

Several similar situations can be found but in cases where the initial and goal configurations are well arranged (e.g. initial and goal orientations are nearly parallel to the walls as usual in everyday car-parking and manuvering tasks) we experienced a great reliability of the planning algorithm.

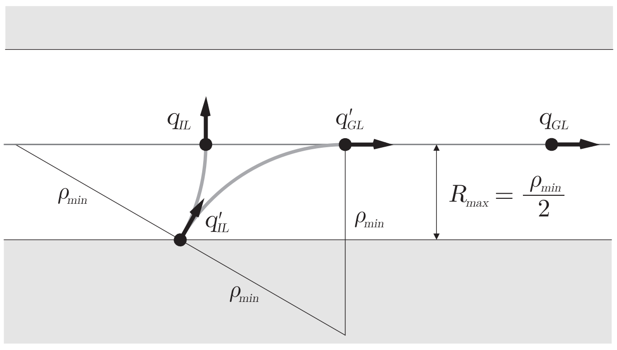


Fig. 8. A boundary situation where the algorithm almost fails

The chance of finding a solution is influenced by the resolutions k_x and k_y as well. We experienced that in case of long narrow corridors a solution could be found reliably if the corridor width was not smaller than $3 \cdot \max(\Delta x, \Delta y)$, where Δx and Δy stand for the grid units in x and y directions, respectively.

7 Conclusions

A novel approach to path planning for a point robot with car-like kinematics was presented in this paper. The planner obtains a feasible path as a concatenation of $SC(S)/CC(S)$ local paths. A preliminary global geometric path is planned in advance by triangular cell decomposition of the free space, which guides the local planner module to find goal-directed local feasible paths.

The planning algorithm has some limitations which motivate further work in this topic. The most important one is the need for extending the algorithm to polygonal vehicle shapes. Another issue of the current implementation is that local intermediate goal orientations $\theta\left(q'_{GL}\right)$ obtained by the LPM are not forced to coincide exactly with $\theta(q_{GL})$ just to be parallel to the line of q_{GL}. This means that both orientations $\theta\left(q'_{GL}\right) = \theta(q_{GL})$ and $\theta\left(q'_{GL}\right) = \theta(q_{GL}) + \pi$ are accepted. This can cause long backward motions, and in addition to that a final orientation opposed to the original q_G of the planning query is possible as well.

Since the LPM looks for the shortest one among the candidate $SC(S)/CC(S)$ paths, this can result in solutions that lead the robot quite close to obstacles (such path segments can be seen e.g. in Figure 5 and Figure 7). Solutions with more safety could be obtained with an LPM maximizing the path clearance instead of minimizing its length. Of course, this would result in higher computational complexity.

Another direction of future work is the extension of the proposed method to more realistic car models. While the LPM presented in this paper generates feasible paths for the simple kinematic model (3), these paths are not feasible for a real car, since (3) allows an abrupt change in the steering angle. In order to respect the limited steering rate of real cars, continuity of the path curvature and an upper bound on the curvature derivative have to be ensured. This can be accomplished by incorporating continuous curvature path segments, e.g. clothoid arcs into the set of path primitives of the local planner module.

References

1 **Latombe J-C**, *Robot Motion Planning*, Kluwer Academic Publishers; Boston, MA, 1991.

2 **Laumond J-P**, *Robot Motion Planning and Control*, Lecture Notes in Control and Information Sciences, Vol. 229, Springer, 1998, ISBN 3-540-76219-1.

3 **LaValle SM**, *Planning Algorithms*, Cambridge University Press; Cambridge, U.K., 2006. Available online at `http://planning.cs.uiuc.edu/`.

4 **B. Siciliano and O. Khatib (Eds.)**, *Springer Handbook of Robotics*, Springer-Verlag, 2008, ISBN 978-3-540-23957-4.

5 **Lafferriere G, Sussmann H**, *Motion planning for controllable systems without drift*, In: Proceedings of the IEEE International Conference on Robotics and Automation, 1991, pp. 1148-1153, DOI 10.1109/ROBOT.1991.131763.

6 **Murray RM, Sastry SS**, *Nonholonomic motion planning: steering using sinusoids*, IEEE Trans. Autom. Control, **38**, (1993), 700-716, DOI 10.1109/9.277235.

7 **Sekhavat S, Laumond J-P**, *Topological property for collision-free nonholonomic motion planning: the case of sinusoidal inputs for chained form systems*, IEEE Trans. Robot. Autom., **14**, (1998), 671-680, DOI 10.1109/100.580977.

8 **Tilbury D, Murray RM, Sastry SS**, *Trajectory generation for the N-trailer problem using Goursat normal form*, IEEE Trans. Autom. Control, **40**, (1995), 802-819, DOI 10.1109/9.384215.

9 **Fliess M, Lévine J, Martin P, Rouchon P**, *Flatness and defect of nonlinear systems: Introductory theory and examples*, International Journal of Control, **61**, (1995), 1327-1361, DOI 10.1080/00207179508921959.

10 **Fernandes C, Gurvits L, Li ZX**, *A variational approach to optimal nonholonomic motion planning*, In: Proceedings of the IEEE International Conference on Robotics and Automation, 1991, pp. 680-685, DOI 10.1109/ROBOT.1991.131662.

11 **Dubins LE**, *On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents*, American Journal of Mathematics, **79**, (1957), 497-516.

12 **Reeds JA, Shepp LA**, *Optimal paths for a car that goes both forwards and backwards*, Pacific Journal of Mathematics, **145**, (1990), 367-393.

13 **Sussmann HJ, Tang G**, *Shortest Paths For The Reeds-Shepp Car: A Worked Out Example Of The Use Of Geometric Techniques In Nonlinear Optimal Control*, Department of Mathematics, Rutgers University, September, 1991, Report No.:SYCON-91-10.

14 **Giordano PR, Vendittelli M, Laumond J-P, Souères P**, *Nonholonomic Distance to Polygonal Obstacles for a Car-Like Robot of Polygonal Shape*, IEEE Trans. Robot., **22**, (2006), 1040-1047, DOI 10.1109/TRO.2006.878956.

15 **P. Souères P, Laumond J-P**, *Shortest paths synthesis for a car-like robot*, IEEE Trans. Autom. Control, **41**, (1996), 672-688, DOI 10.1109/9.489204.

16 **Balkcom DJ, Mason MT**, *Time Optimal Trajectories for Bounded Velocity Differential Drive Vehicles*, International Journal of Robotics Research, **21**, (2001), 199-217, DOI 10.1177/027836402320556403.

17 **Chitsaz H, O'Kane JM, Balkcom DJ, Mason MT**, *Minimum wheel-rotation paths for differential-drive mobile robots*, In: Proceedings of the IEEE International Conference on Robotics and Automation, 2006, pp. 1616-1623, DOI 10.1109/ROBOT.2006.1641938.

18 **Kavraki LE, Svetska P, Latombe J-C, Overmars MH**, *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, IEEE Trans. Robot. Autom., **12**, (1996), 566-580, DOI 10.1109/70.508439.

19 **LaValle SM**, *Rapidly-exploring random trees: A new tool for path planning*, Computer Science Dept., Iowa State University, 1998.

20 **Laumond J-P, Jacobs PE, Taïx M, Murray RM**, *A Motion Planner for Nonholonomic Mobile Robots*, IEEE Trans. Robot. Autom., **10**, (1994), 577-593, DOI 10.1109/70.326564.

21 **Mirtich B, Canny J**, *Using skeletons for nonholonomic path planning among obstacles*, In: Proceedings of the IEEE International Conference on Robotics and Automation, 1992, pp. 2533-2540, DOI 10.1109/ROBOT.1992.220060.

22 **Sekhavat S, Chyba M**, *Nonholonomic deformation of a potential field for motion planning*, In: Proceedings of the IEEE International Conference on Robotics and Automation; Detroit, MI, 1999, pp. 817-822, DOI 10.1109/ROBOT.1999.770075.

23 **Kim J, Chung W, Park S**, *Practical motion planning for car-parking control in narrow environment*, IET Control Theory and Applications, **4**, (2010), 129-139, DOI 10.1049/iet-cta.2008.0380.

24 **Ghita N, Kloetzer M**, *Trajectory planning for a car-like robot by environment abstraction*, Robotics and Autonomous Systems, **60**, (2012), 609-619, DOI 10.1016/j.robot.2011.12.00.

25 **Kloetzer M, Ghita N**, *Software tool for constructing cell decompositions*, In: Proceedings of the IEEE Conference on Automation Science and Engineering (CASE); Trieste, Italy, 2011, pp. 507-512, DOI 10.1109/CASE.2011.6042492.

26 **Minguez J, Montano L**, *Extending Collision Avoidance Methods to Consider the Vehicle Shape, Kinematics, and Dynamics of a Mobile Robot*, IEEE Trans. Robot., **25**, (2009), 367-381, DOI 10.1109/TRO.2009.2011526.

27 **Liu Y-H, Arimoto S**, *Path Planning using a Tangent Graph for Mobile Robots among Polygonal and Curved Obstacles*, International Journal of Robotics Research, **11**, (1992), 376-382, DOI 10.1177/027836499201100409.

28 **Ó'Dúnlaing C, Yap CK**, *A "retraction" method for planning the motion of a disc*, Journal of Algorithms, **6**, (1985), 104-111, DOI 10.1016/0196-6774(85)90021-5.

29 **Shewchuk JR**, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, In: **Ming C. Lin and Dinesh Manocha** (ed.), Applied Computational Geometry: Towards Geometric Engineering, Lecture Notes in Computer Science, Vol. 1148, Springer-Verlag, May 1996, pp. 203-222.

30 **Shewchuk JR**, *Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator*, 2005. `http://www.cs.cmu.edu/~quake/triangle.html`.

31 **Dijkstra EW**, *A Note on Two Problems in Connexion with Graphs*, Numerische Mathematik, **1**, (1959), 269-271.

32 **Fredman ML, Tarjan RE**, *Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms*, Journal of the Association for Computing Machinery, **34**(3), (1987), 596-615, DOI 10.1145/28869.28874.