Robotics and Autonomous Systems 79 (2016) 1–11

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Optimisation based path planning for car parking in narrow environments

Patrik Zips *, Martin Böck, Andreas Kugi

Automation and Control Institute, Vienna University of Technology, 1040 Vienna, Austria

HIGHLIGHTS

- Path is calculated by solving several local static optimisation problems iteratively.
- Driving direction changes are locally determined based on simple rules.
- Obtained paths are similar to manoeuvres of a human driver.
- Tree-based extension guides the local planner efficiently.
- High rate of success in narrow environments with low computing costs.

ARTICLE INFO

Article history:
Received 2 March 2015
Received in revised form
17 February 2016
Accepted 26 February 2016
Available online 4 March 2016

Keywords:
Non-holonomic path planning
Car-like robots
Static optimisation
Minkowski sum

ABSTRACT

In the last decades, a large variety of robot motion planners emerged. However, manoeuvring in very narrow environments, e.g., for common parking scenarios, cannot be reliably handled with existing path planners at low computing costs. This is why, this paper presents a fast optimisation based path planner which specialises on narrow environments. In the proposed approach, the kinematic differential equations are discretised. For the resulting discrete path segments, a static optimisation problem is formulated to determine the path independently of the considered scenario. In each iteration step, the path length is also optimised to handle close distances to the obstacles as well as longer driving distances. Due to the local nature heuristic rules for driving direction changes are formulated which intend to imitate the behaviour of a human driver. The drawback of the local nature is the lack of global information to handle scenarios with obstacles blocking the path. To overcome this problem, a tree-based guidance for the local planner is introduced. The landmarks for the tree are implicitly created by the local planner allowing an efficient exploration of the free configuration space. The performance of the algorithm is evaluated utilising Monte-Carlo simulations and compared to state-of-the-art path planning algorithms.

1. Introduction

One prevailing topic of recent research in automotive industry is autonomous driving, which is especially challenging in urban environments. There, narrow corridors, tight turns, and the partly unpredictable behaviour of other traffic participants have to be handled in a systematic way. A special topic within this area of research is automated parking control, which is not only useful for autonomous vehicles, but also in conventional cars as parking assistance system. A necessary requirement for such a system is a path planner which plans a feasible path between the current position and a specified parking position. Hereby, the challenge arises from the non-holonomic constraints of the car and the proximity to the obstacles.

In robot path planning, several algorithms have been developed to tackle this problem. One class of path planners rely on building up a roadmap in the considered environment. The map is set up in a learning phase at start-up and saved as a graph with the landmarks as nodes and the feasible paths as edges. After the learning phase, the path planning can be performed in a very efficient way by using this graph. A famous representative of this class is the so called kinematic roadmap. Based on "Ariadne's Clew" [1,2], it consists of a global planner EXPLORE and a local planner SEARCH. The global planner places landmarks in the reachable region of an already existing landmark and the local planner tries to connect these placed landmarks with the target position. All landmarks together are called kinematic roadmap. A similar approach is the "probabilistic roadmap" [3]. Here, landmarks are randomly placed

* Corresponding author. Tel.: +43 158801 376260; fax: +43 158801 37699.
E-mail address: zips@acin.tuwien.ac.at (P. Zips).

http://dx.doi.org/10.1016/j.robot.2016.02.004
0921-8890/

in the free configuration space based on a certain probability distribution. In this approach, sensor uncertainties can be explicitly considered, e.g., by utilising Monte-Carlo simulations to obtain a probability of success for each edge of the graph [4].

Another class of methods builds up a tree of robot motions beginning from the starting position. In this context, the most popular representative are the Rapidly-exploring Random Trees (RRT) [5]. Here, the point from where the tree expansion starts is chosen close to the target position in each iteration. This size of the Voronoi regions together with a probability distribution serves as decision criterion. To account for kinematic constraints of a robot, the tree can be expanded using a constant input for a certain driving distance. For example, randomised inputs [6] and inputs based on the analytic solution [7] are used. Integrating the differential equations of the robot for a specified distance with the chosen input yields a new configuration (also called node) from which the process is repeated. If a node is in collision, it is discarded and an existing node in the tree is picked as a new starting point. In this manner, a path can be iteratively constructed.

Path planning for cars and car-like robots received special attention in robot motion planning. By focusing on these systems, improved convergence and path quality can be achieved. The first works in this field trace back to Dubins, who analysed the paths for a particle with constrained curvature [8]. It is shown that the shortest path for this particle in the (x, y)-plane is composed of lines and arcs with minimal curvature. These ideas can be directly applied to a car, as the non-holonomic and the physical constraints result in a constrained curvature. Reeds and Shepp generalised these results for a car which drives forwards and backwards [9]. They defined nine curvature classes, which can describe every optimal path between any points in the (x, y)-plane considering the orientation. However, in [8,9] only obstacle free environments are considered. Therefore, many algorithms combine these ideas with a suitable obstacle avoidance scheme. For example, a path into a parking spot can be obtained by stringing together lines and minimal curvature arcs, see, e.g., [10–12]. In [13] these lines and arcs are defined as translation (TM) and rotation movements (RM). Starting from the parking position, N-cycles of every possible TM–RM–TM combination are linked together, where N has to be chosen according to the available computing time and the problem complexity. A higher number of iterations N consequently leads to a larger computing time. Instead of lines and arcs other curves can be used to obtain smoother paths, e.g., β-spline [14,15], Beziér [16] or polynomial [17] curves.

Another method based on Reeds and Shepp's curves combined with the well-known A*-algorithm is proposed in [18]. Here, the system equations are integrated for a certain distance with a constant input. The continuous end configuration of the integration is assigned to a discrete cell. A heuristics relying on Reeds and Shepp's curves and a holonomic path finder guides the path planner towards the target position. The input of the system also includes the driving direction which allows to calculate driving direction switching points. A cost penalty for changing the driving direction aims at minimising the overall number of driving direction switching points.

To handle the non-holonomic characteristics of a car, [19] proposes an algorithm which computes a holonomic path for a given environment and tries to move along this path under consideration of the non-holonomic constraints. Due to the fact that a car is small-time-controllable [20] this is always possible, but often results in highly manoeuvring paths particularly in narrow environments. Thus, different methods to follow a holonomic path are proposed. In [21] the differential equations of the car are transformed into a chained-form system and sinusoidal inputs are applied. In [22] Dubins curves are used to obtain the non-holonomic path and in [23] a local continuous curvature planner using clothoids, as presented in [24], in combination with a shortest feasible path metric [25] is used.

Other approaches rely on solving an optimal control problem. Here, the challenge arises from the mathematical description of the obstacles. In [26] the obstacles are discretised to obtain boundaries composed of a finite number of points. Every point has a potential field value which increases if the vehicle gets closer. The values of all points are summed up in one inequality constraint which has to be lower than or equal to zero. This inequality can only hold if no point collides with the vehicle. The drawback of this method is the inaccurate description of the obstacles. By adding more points, the model gets more precise but the computing time increases.

The described path planners, among others, are able to calculate feasible paths for most environments. Nevertheless, no algorithm yields satisfactory results for typical car parking which consists of the parking at a parking spot as well as parking scenarios at a parking deck or in a narrow parking lot. The methods of classical robot path planning, like the roadmap and the tree-based methods, rely on discretising the available space or assigning a certain distance for integration. Thus, in narrow environments the discretisation has to be decreased which largely increases the computing costs for longer driving distances. Methods specialised on car parking are able to obtain paths in narrow environments in an efficient way for one certain scenario, like, e.g., parallel parking. However, they cannot calculate feasible paths in more general environments, like, for instance, a parking deck.

This is why, an optimisation based path planner specialised on narrow environments with low computational costs for real-time applications is proposed. The step length is variable and included in the optimisation problem. Thus, the planner is able to handle longer driving distances in narrow environments without increasing the computing costs. The cost function is designed to guide the path planner towards the target position and does not rely on a specific scenario. The paper is organised as follows: Section 2 introduces the considered car model and environment. Section 3 presents the path planning concept for common parking spots by solving a sequence of static optimisation problems. Section 4 is devoted to an extension of the parking algorithm to handle longer driving distances, as they typically appear on a parking deck, by calculating a tree of landmarks. Simulation studies for different parking scenarios are carried out in Section 5 showing the practical feasibility of the proposed algorithm. Section 6 contains some conclusions of the presented path planner. This paper extends our previous work [27] giving insight into the choice of the parameters and improving the convergence of the path planner. Additionally, a tree-based extension is proposed to guide the local path planner for longer driving distances. Furthermore, quantitative results of the convergence are provided by extensive Monte-Carlo simulation studies and compared to different state-of-the-art path planners.

2. Problem statement

Three common parking scenarios, namely parallel, garage, and angle parking, constitute the starting point of this paper. These scenarios and the corresponding notations are shown in Fig. 1, where the shaded polygons represent obstacles like, e.g., other cars. The lines to the left and right side of each scenario are boundaries which shall not be violated, e.g., the side walk or the lane separator of the street. In a further step, car parking in more general environments as, for instance, a parking deck and a parking lot, will be considered.

2.1. System dynamics

In the following, the car model with Ackerman steering, as depicted in Fig. 2, serves as a basis for the mathematical description of the car. Thereby, the tyre slip angle is neglected, which is

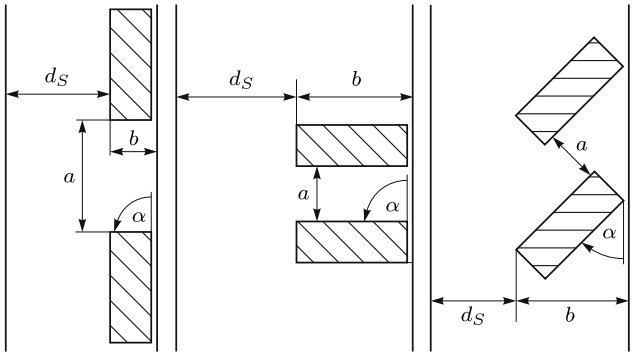


Fig. 1. Notation for parallel, garage and angle parking.

justified by the low velocities during parking manoeuvres. Hence, the car can be described by one front and one rear wheel. The configuration and motion of the car are characterised by the coordinates (x, y) of a reference point P_R, which is located at the centre of the rear axle, the orientation θ of the longitudinal axis, the velocity v of the car and the steering angle δ. The non-holonomic differential equations in these generalised coordinates $\tilde{\mathbf{q}} = [x, y, \theta, v, \delta]^{\mathrm{T}}$ read as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x \\ y \\ \theta \\ v \\ \delta \end{pmatrix} = \begin{pmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{L}\tan(\delta) \\ a \\ \zeta \end{pmatrix} = \mathbf{f}_t(\tilde{\mathbf{q}}, \mathbf{u}_A). \tag{1}$$

Here, $\mathbf{u}_A = [a, \zeta]^{\mathrm{T}}$ describes the control input with the acceleration a and the steering angle speed ζ, and the parameter L refers to the wheelbase. The kinematic path planning can be decoupled from the dynamic planning problem by Path–Velocity-Decomposition [28]. To this end, the velocity can be written as $v = D\frac{\mathrm{d}s}{\mathrm{d}t}$, with the path position s. Thereby, $D \in \{-1, 1\}$ refers to the driving direction of the car, with $D = 1$ for velocities $v \geq 0$ and $D = -1$ for $v < 0$. Thus, the kinematic state of the car $\mathbf{q} = [x, y, \theta]^{\mathrm{T}}$ can be obtained from (1) and reads as

$$\mathbf{q}' = \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} D\cos(\theta) \\ D\sin(\theta) \\ Du_l \end{pmatrix} = \mathbf{f}_s(\mathbf{q}, u_l, D) \tag{2a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}_d = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} v \\ \delta \end{pmatrix} = \begin{pmatrix} a \\ \zeta \end{pmatrix} = \mathbf{f}_d(v, \mathbf{u}_A) \tag{2b}$$

where $(\cdot)'$ denotes the derivative with respect to s. The system is now split up into the kinematic state equation (2a) with the new control input $u_l = \frac{\tan(\delta)}{L}$ and D and the dynamic state equation (2b). For the path planning, only the kinematic equations (2a) are considered simplifying the problem resulting in lower computational costs. The dynamic state describes the velocity to traverse the kinematic path. Although rapid changes in the steering angle can be realised at lower velocities (or stopping the car) they have to be avoided to increase driving comfort. Thus, a path is considered feasible if the steering angle is sufficiently smooth. This is realised using the cost function of the optimisation problem as explained in the sequel.

2.2. Path planning problem

The path planner has to find a feasible path for the car from a given starting configuration $\mathbf{q}_S$ to a parking configuration $\mathbf{q}_P$. The path shall have a reasonable length and must not collide with any obstacle or exceed a boundary. Moreover, the same path planner shall be able to plan a path for all considered scenarios.

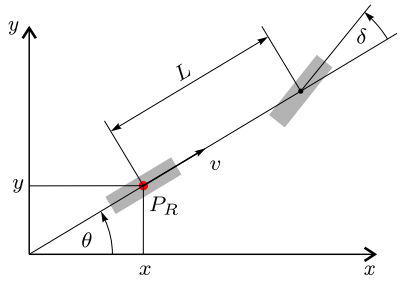


Fig. 2. Kinematic model of the car.

Without loss of generality, the obstacles are presumed to have a convex polygonal shape. At cost of higher computing time, every non-convex polygon can be subdivided into multiple convex polygons, which is usually referred to as convex decomposition, see, e.g., [29]. Non-polygonal obstacles can always be included in a slightly larger polygon. For collision checking, the actual boundary of a car is approximated by a convex polygon with 14 corners.

3. Path planning for car parking

In a first step, path planning for typical parking spots, see Fig. 1, is considered. To this end, the differential equation (2a) is discretised with respect to the path parameter s by means of a Runge–Kutta discretisation of second order with a step length $\eta_i \in [\eta_{\min}, \eta_{\max}]$. Neglecting the error term of the discretisation yields the difference equation

$$\mathbf{q}_{i+1} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + D\eta_i \cos\left(\theta_i + D\frac{\eta_i u_{l_i}}{2}\right) \\ y_i + D\eta_i \sin\left(\theta_i + D\frac{\eta_i u_{l_i}}{2}\right) \\ \theta_i + D\eta_i u_{l_i} \end{pmatrix} = \mathbf{f}(\mathbf{q}_i, \mathbf{u}_i, D). \tag{3}$$

The new control input $\mathbf{u} = [u_l, \eta]^{\mathrm{T}}$ consists of the steering input u_l and the step length η. For better readability, the index of D_i will be neglected in the following as the driving direction is typically constant over multiple optimisation steps.

Based on the difference equation (3) a series of static optimisation problems is formulated to calculate the path. The optimisation problem does not calculate the whole path at once but only for one incremental step. This procedure is then applied in a recursive manner and based on the local behaviour in each iteration it is decided whether a direction change of the car is deemed necessary. The next subsection will detail how these switching points are chosen by the algorithm. The constrained static optimisation problem to be solved in each iteration takes the form

$$\min_{\mathbf{u}_i} \; l_{O_i}(\mathbf{q}_{i+1}) \tag{4a}$$

$$\text{s.t. } \mathbf{q}_{i+1} = \mathbf{f}(\mathbf{q}_i, \mathbf{u}_i, D) \tag{4b}$$

$$\mathbf{h}_P(\mathbf{q}_{i+1}) \leq \mathbf{0} \tag{4c}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_i \leq \mathbf{u}_{\max}, \tag{4d}$$

with the cost function

$$l_{O_i}(\mathbf{q}_{i+1}) = r_\theta e^2_{\theta_{i+1}} + \mathbf{e}^{\mathrm{T}}_{P_{i+1}}\mathbf{R}\mathbf{e}_{P_{i+1}} + r_u \Delta u_i^2. \tag{5}$$

Thereby, $\mathbf{e}_{P_i} = [x_i - x_E, y_i - y_E]^{\mathrm{T}}$ denotes the distance of the car at iteration step i to the desired target configuration of the path $[x_E, y_E]^{\mathrm{T}}$ and $e_{\theta_i} = \theta_i - \theta_O$ refers to the difference between the car orientation θ_i and a predefined target angle θ_O. The parameter $r_\theta > 0$ and the positive semi-definite matrix $\mathbf{R}$ serve as

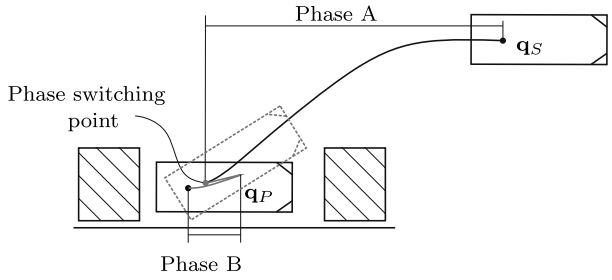


Fig. 3. Example for the phases at a parallel parking spot.

weighting terms in the cost function. The steering input difference $\Delta u_i = u_{l_i} - u_{l_{i-1}}$ together with the weighting term r_u minimises the steering input change in each iteration yielding a smooth steering angle along the path. The constraint (4b) corresponds to (3) and (4c) accounts for collision avoidance using the Minkowski sum [30]. The control input $\mathbf{u}_i$ is constrained by $\mathbf{u}_{\min} = [-u_{l_{\max}}, \eta_{\min}]^{\mathrm{T}}$ and $\mathbf{u}_{\max} = [u_{l_{\max}}, \eta_{\max}]^{\mathrm{T}}$.

In every iteration step, the position and the orientation of the car is improved with respect to the cost function (5). The weighting terms r_θ, $\mathbf{R}$ and r_u of the cost function as well as the target angle θ_O are used to induce a certain behaviour of the path planner. To determine suitable weighting terms for the cost function (5) as well as the target angle θ_O, the parking algorithm is separated into two phases which correspond to two different tasks. The first phase, henceforth referred to as phase A, is responsible for steering the car to a vicinity of the parking spot. For this, a suitable path between two predefined points must be found, mostly with few driving direction switching points. Phase B handles the parking manoeuvre itself, where typically small but precise position and orientation changes have to be achieved. The change between phases A and B is characterised by the so called phase switching point, which will be explained in more detail later in this section. The path planning is carried out in backward direction starting from the parking configuration in phase B. It continues via the phase switching point to the starting configuration in phase A. Thus, the initial configuration for the path planner is set to $\mathbf{q}_0 = \mathbf{q}_P$ and the target configuration to $\mathbf{q}_E = \mathbf{q}_S$. An example for these phases at a parallel parking spot is depicted in Fig. 3. Here, the driving direction is changed once in phase B. The phase switching point is depicted as dashed contour.

The first task is to find a suitable path from the parking configuration to the phase switching point, from where it is ensured that the car is able to leave the parking spot. For garage and angle parking it can be reached by driving straight forward or backward. For parallel parking, changing the orientation of the car leads to a path to the phase switching point. Therefore, the car orientation θ is most important whereas the weighting matrix $\mathbf{R}$ can be set to zero. The target angle θ_O depends on the respective parking scenario. For garage and angle parking, the car orientation shall be held constant and for left and right parallel parking decreased or increased, respectively. A target angle θ_O which can be used for all scenarios is the modified parking orientation $\theta_O = \theta_P + \gamma$ with

$$\gamma = \begin{cases} \pm\dfrac{\pi}{2} & \text{for parallel parking} \\ 0 & \text{else,} \end{cases} \tag{6}$$

where the angle γ is positive for parallel parking spots to the right and negative for spots to the left.

In phase A, the path planner has to find a suitable path from the phase switching point to the starting configuration. To reach the starting configuration the non-holonomic constraints of the car have to be respected, i.e., it is easy to drive forwards and backwards but a sidewards movement is involved with high effort. This property can be taken into account by suitably choosing the weighting terms r_θ and $\mathbf{R}$ in (5). To this end, the coordinate system is subsequently defined to have its origin in the reference point P_R at the starting configuration $\mathbf{q}_S$. The x-axis coincides with the longitudinal axis of the car. This implies that the x-direction at the starting configuration is in driving direction and consequently the y-direction is orthogonal to it. Thus, close to the starting configuration deviations in the y-direction can be directly attributed to a high effort for movements correcting this deviation. Additionally, deviations of the orientation at the starting configuration also result in a high effort as the car cannot turn without changing its position. In order to minimise this final effort, the y- and θ-deviations have to be sufficiently small in the neighbourhood of the starting configuration. To this end, a higher value is chosen for the weighting term corresponding to the deviation in y-direction followed by the weighting term for the θ-deviation. The lowest value is assigned to the weighting of the x-deviation. The target angle is set equal to the starting angle $\theta_O = \theta_S = 0$. These weighting terms ensure that the deviations associated with a high effort are minimised and that the starting configuration can be reached by driving straight for- or backwards.

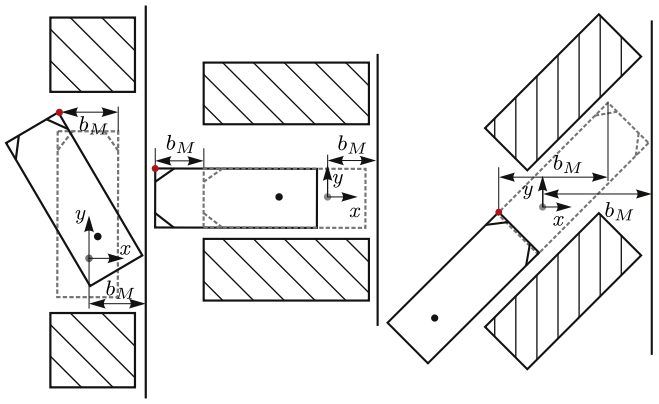


Fig. 4. Phase switching point for (I) parallel, (II) backward garage and (III) angle parking for right parking spots.

3.1. Switching points

This subsection is devoted to the determination of switching points between phases A and B and for driving direction changes. Firstly, the phase switching point is treated. From the phase switching point, the car shall be able to leave the parking spot and drive towards the starting configuration. A suitable point for all three scenarios is when the front corner of the car moves a sufficiently large distance b_M in x-direction in a coordinate system tied to the parking configuration, see Fig. 4. Assuming a common parking configuration, a suitable distance b_M is the distance between the boundary at the end of the parking spot to the reference point of the car at the parking configuration. This choice also avoids too early steering inside a garage or angle parking spot which may lead to close distances to the obstacles bordering the parking spot. Fig. 4 shows an example for a parallel, backward garage and angle parking spot. Here, the parking configuration is depicted as dashed contour and the phase switching point as solid contour. This heuristics can be formulated as an inequality condition $x_{C_{FL}} \geq b_M$ for parking spots to the left and $x_{C_{FR}} \leq b_M$ for parking spots to the right. Thereby, $x_{C_{FL}}$ and $x_{C_{FR}}$ describe the x-position of the front left and front right corner of the car, respectively, in a coordinate system tied to the parking configuration.

Switching points for driving directions are calculated using heuristic rules based on the local behaviour of the path planner.

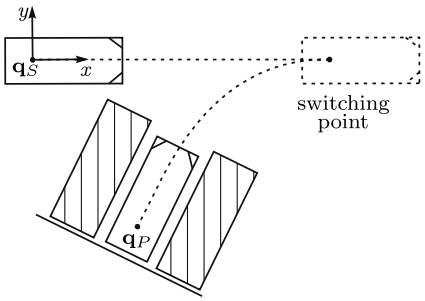


Fig. 5. Vehicle needs to switch direction while driving to the parking spot.

The first rule, mainly concerning phase B, is to switch direction if there is not enough free space in driving direction to make a step with the minimum step length $\eta_{\min}$. This means that the optimisation problem (4) has no feasible solution subject to the constraints. At this point the direction is switched, i.e., D in (3) is set to $-D$, and the optimisation problem (4) is solved again. If this again yields no solution, the path planner is not able to find a feasible path between the starting and the parking configuration.

The second rule is primarily designed for phase A. Here, driving direction switching points may be important for the car to reach the starting configuration. An example is shown in Fig. 5 for garage parking. The starting configuration cannot be reached from the parking configuration in one draw. The best way is to drive out of the parking spot to the right and change direction as soon as the y- and θ-deviation from the starting configuration is sufficiently small. At this point, the second purpose of the choice of the weighting terms for phase A gets evident. If the y-position and the orientation angle θ are close enough to the starting configuration the desired x-position can be easily reached by driving straight for- or backwards. In optimisation terms, the cost function will decrease until the θ- and y-deviation from the starting configuration are sufficiently small and then increase again with increasing x-deviation. Thus, the driving direction is switched if $l^*_{O_i} > l^*_{O_{i-1}}$ and the optimisation problem (4) solved again for D replaced by $-D$. The superscript * refers to the optimal values, i.e., $l^*_{O_i} = l_{O_i}(\mathbf{q}^*_{i+1})$ and $\mathbf{q}^*_{i+1} = \mathbf{f}(\mathbf{q}_i, \mathbf{u}^*_i, D)$.

The condition for the phase switching point and the second rule for driving direction changes are the reasons why the path is planned backwards. If the path planning were carried out in forward direction, coordinates for all switching points would have to be defined in advance instead of using these two simple heuristic rules.

3.2. Algorithm

The parking algorithm is summarised in Algorithm 1. Here, a path from an initial configuration $\mathbf{q}_0$ to a target configuration $\mathbf{q}_E$ is planned. This notation will be used in the next section. A parking spot on the right side is chosen, whereby r_{θ_A} and $\mathbf{R}_A$ denote the weighting terms for phase A. For a parking spot to the left the **if** condition in line 4 has to be changed to $x_{C_{FL},i} \geq b_M$, where the index i refers to the current configuration $\mathbf{q}_i$. The second rule for direction changing $l^*_{O_i} > kl^*_{O_{i-1}}$ is extended by a factor $1 \leq k < 2$ to allow for longer driving distances in one direction reducing the overall number of driving direction changes. The algorithm stops if the current position is in an ϵ-neighbourhood of the target configuration or if the number of iterations i or direction changes c reach their maximum value i_M or c_M, respectively.

Algorithm 1 Parking Algorithm.

1: $r_\theta = 1, \mathbf{R} = \mathbf{0}$ {weighting terms}
2: $l_{O_0} = \infty, i = 0, c = 0$
3: **repeat**
4: **if** $x_{C_{FR},i} \leq b_M$ **and** in phase B **then**
5: {switch to phase A}
6: $\theta_O = \theta_E, r_\theta = r_{\theta_A}, \mathbf{R} = \mathbf{R}_A$
7: **end if**
8: Solve (4) to obtain $\mathbf{u}^*_i$
9: **if** $l^*_{O_i} > kl^*_{O_{i-1}}$ **or** any $h_{P_j}(\mathbf{q}^*_{i+1}) > 0$ **then**
10: $D = -D$
11: $c \leftarrow c + 1$
12: **else**
13: $\mathbf{q}_{i+1} \leftarrow \mathbf{f}(\mathbf{q}_i, \mathbf{u}^*_i, D)$
14: $i \leftarrow i + 1$
15: **end if**
16: **until** $||\mathbf{q}_i - \mathbf{q}_E|| \leq \epsilon$ **or** $i > i_m$ **or** $c > c_M$

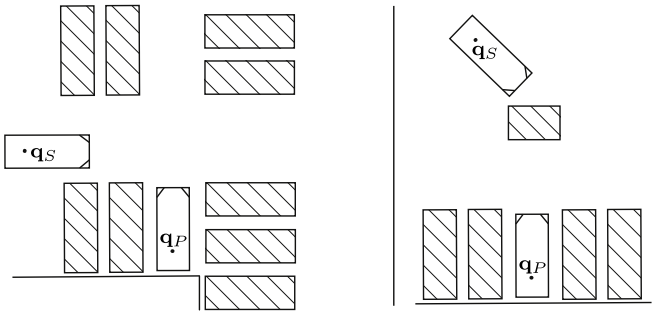


Fig. 6. Parking scenario at a parking deck and on a parking lot.

4. Path planning in narrow environments

The parking algorithm described in the previous section is mainly efficient for common parking scenarios with starting configurations along a street next to the parking spot. If the starting configuration is farther away, the proposed optimisation based path planner may not find a path around large obstacles or may run into dead ends, mainly due to its local nature. To deal with these problems, an extension is proposed which guides the local path planner towards the starting configuration. The aim is to effectively handle a larger variety of parking scenarios, e.g., parking at a parking deck or at a parking lot as depicted in Fig. 6. These scenarios are characterised by very narrow environments. The starting configuration may be at an arbitrary position in the free configuration space and additional obstacles may block the shortest path between the starting and the parking configuration.

4.1. Tree-based global guidance

In robot motion planning, a common approach is to utilise trees with landmarks to handle complex scenarios, see Section 1 for some examples. Path planners for holonomic systems usually create random landmarks based on heuristic rules and connect these landmarks with a local path planner. For non-holonomic systems, placing landmarks is a difficult task as these landmarks have to be collision free and reachable from another landmark or from the initial configuration. Furthermore, landmarks closer to the target configuration in the sense of an Euclidean metric do not necessarily correspond to an improved configuration (with respect to the non-holonomic constraints). Thus, the validation of newly created landmarks is a non-trivial task. Because of these reasons, finding landmarks for arbitrary scenarios is computationally challenging with high demands regarding the analysis of the free space.

To avoid searching for landmarks, a common way is to integrate the underlying differential equations with a constant input for a specific path length [6,18]. The end point of the integration is used as a new landmark provided that the path is collision free. The path planner is guided to the target configuration using heuristic rules by picking the landmark from where to start the next integration. In general, this leads to a high number of landmarks. Reducing the number of landmarks, e.g., by longer integration path lengths or matching them with pre-sampled cells [18], lowers the computational costs. The drawback of the reduction is worse convergence of the path planner in particular in narrow environments.

In the following, a different approach for finding suitable landmarks spanning a tree is proposed. The landmarks are created implicitly by the local path planner during the planning phase. The local planner is utilised to plan paths to specific reference configurations. These reference configurations are chosen based on heuristic rules which are tailored to common urban parking scenarios. Hence, the calculation of the reference configurations is very cheap. A random element is included which allows an efficient exploration of the free configuration space with low computational costs. The local planner ensures a good coverage of the configuration space with landmarks close to the obstacles. For all landmarks, suitable costs based on the path quality to this landmark and the chance of success for the path planning are assigned. These quality criteria are used to expand the tree from the landmark with the lowest costs. Simulation studies will show a clear improvement compared to the state of the art in particular for narrow environments, cf. Section 5.

4.2. Reference configurations

First, suitable candidates for the reference configurations will be discussed. As mentioned before, the optimisation based local path planner presented in the previous section is used to plan from an initial configuration $\mathbf{q}_0$ to a reference configuration $\mathbf{q}_R$ which serves as an intermediate target configuration $\mathbf{q}_E$. This reference configuration is obtained using simple heuristic rules to decrease the computational costs. The requirements on the reference configuration are very low; it neither has to be collision free nor reachable. An already reached landmark or the parking configuration itself may serve as initial configuration, as will be explained in the next subsection.

The heuristic rules to calculate the reference configurations are obtained by investigating common urban parking scenarios. Thereby, the focus is on garage parking which constitutes the prevailing scenario on parking decks and parking lots. The considered environments are usually composed of (nearly) straight lanes which are arranged in parallel and orthogonal to each other. Thus, a car in such an environment mostly drives straight ahead along a lane or turns orthogonal into another lane. Also driving into and out of a parking spot corresponds to an orthogonal turn. Based on these considerations, two different classes for the reference configurations are defined as parallel displacement and orthogonal displacement. For the parallel displacement, the reference configuration $\mathbf{q}_{R_P}$ is placed in front of the initial configuration $\mathbf{q}_0$ towards the starting configuration $\mathbf{q}_S$ with the same orientation as the initial configuration. For the orthogonal displacement, the orientation of the reference configuration $\mathbf{q}_{R_O}$ is changed to $\theta_R = \theta_0 \pm \pi/2$ with the orientation θ_0 of the initial configuration. Hence, the reference configuration for both cases can be calculated in a coordinate system aligned with the initial configuration $\mathbf{q}_0$ where the x-axis coincides with the longitudinal axis of the car,

$$\mathbf{q}_{R_P} = \mathbf{q}_0 + \begin{bmatrix} \Delta x + \sigma_{x_P} \\ \sigma_{y_P} \\ \sigma_{\theta_P} \end{bmatrix}, \qquad \mathbf{q}_{R_O} = \mathbf{q}_0 + \begin{bmatrix} \Delta x + \sigma_{x_O} \\ \sigma_{y_O} \\ \pm\dfrac{\pi}{2} + \sigma_{\theta_O} \end{bmatrix}. \tag{7}$$

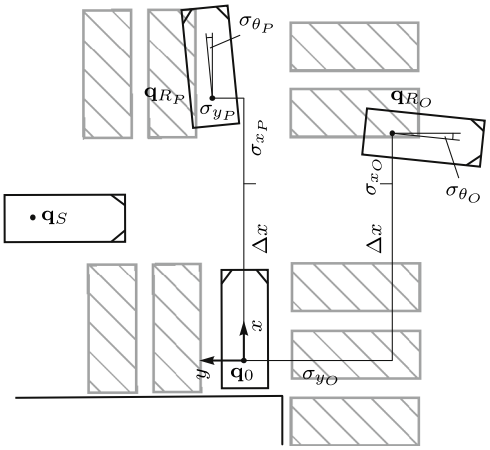


Fig. 7. Two example reference configurations of parallel displacement $\mathbf{q}_{R_P}$ and orthogonal displacement $\mathbf{q}_{R_O}$.

Thereby, $\sigma_{\xi_j}, j \in \{P, O\}$, denotes a uniformly distributed random deviation in the corresponding coordinate $\xi \in \{x, y, \theta\}$. Depending on the direction ξ and the displacement class, the range of σ_ξ is defined. Thus for instance, the random deviation in the angles θ exhibits a small range whereas the deviation in y-direction for orthogonal displacement σ_{y_O} has a larger one. The displacement Δx is a constant parameter to ensure a sufficient large driving distance for the path planning. The $\pm$ sign for the orthogonal angle deviation is chosen to minimise the angle deviation with respect to the starting configuration. Two examples for reference configurations, one of each class, are depicted in Fig. 7. Here, the parking configuration $\mathbf{q}_P$ serves as initial configuration $\mathbf{q}_0$. However, for every other initial configuration the reference configurations are calculated based on the same rules.

Note that the reference configuration neither has to be collision free nor reachable. Its only purpose is to guide the local path planner to explore the free configuration space as explained in the following. The choice of the reference configurations is based on the scenarios under consideration. The path planning framework also allows to incorporate other reference configurations, e.g., obtained by a visual lane detector.

4.3. Path planning using landmarks

Utilising these reference configurations, a tree of connected landmarks is created to find a path between the parking and the starting configuration. To this end, the following procedure is carried out in a sequential manner starting from the parking configuration:

1. Pick a landmark,
2. Calculate two reference configurations and plan to both,
3. Plan from each created landmark to the starting configuration, and
4. Add created landmarks to a sorted list.

In the following, the four steps of this procedure are explained in more detail.

Pick a landmark

The local path planner is started from an initial configuration $\mathbf{q}_0$. Either an already created landmark or the parking configuration qualifies for a suitable initial configuration. The choice is based on a cost sorted list, which will be explained in the sequel. The entry with the lowest cost in this list is used as initial configuration for the local path planner. To avoid picking the same landmark or the parking configuration too often, the costs of the chosen initial configuration are increased by a factor $k_L > 1$.

Calculate two reference configurations and plan to both

Based on the initial configuration two reference configurations, one of each class as described in the previous subsection, are created. Using these reference configurations as intermediate target configuration $\mathbf{q}_E$ the optimisation based local path planner is employed to plan a path according to Section 3. As the reference configuration is neither guaranteed to be collision free nor reachable, the path planner does not have to finish the planning successfully. Instead, the path planning is aborted if either a maximum number of direction switching points c_M is reached or the path is close to the reference configuration. The local path planner places direction switching points based on the rules described in Section 3.1. Hence, each direction switching point is created either because the configuration gets worse with respect to the reference configuration or because there is no feasible step in driving direction. Thus, the direction switching points can also be very close to obstacles.

Each direction switching point as well as the final configuration of the planned path (which can be the last direction switching point and does not necessarily need to be a reference configuration) serves as a landmark $\mathbf{q}_L$ for the further path planning. The landmarks created in this way are guaranteed to be collision free and reachable. Thus, in each planning cycle up to $2(c_M + 1)$ new landmarks can be added to the list, as two paths to two different reference configurations are calculated by the local path planner with up to c_M direction switching points and one final configuration for each path.

Plan from each created landmark to the starting configuration

Each landmark is a suitable candidate for reaching the starting configuration. In complex environments, a strategy to evaluate the likelihood of success for finding a path between a landmark and the starting configuration is computationally demanding, in particular in view of the non-holonomic constraints of the car. Considering the short calculation time of the local path planner, a path is planned from each created landmark to the starting configuration instead. To minimise the overall number of driving direction switching points the landmark with the smallest number of direction changes is used first continuing to the one with the largest number. If the local path planner succeeds, a path from the parking to the starting configuration is found and the path planning is finished.

Add landmarks to sorted list

If the starting configuration is not reached all created landmarks are added to a sorted list based on the cost function

$$l_L = \mathbf{e}_{\mathbf{q}_L}^{\mathrm{T}} \mathbf{R}_L \mathbf{e}_{\mathbf{q}_L} + r_l P_l + r_{SP} N_{SP}. \tag{8}$$

Thereby, $\mathbf{e}_{\mathbf{q}_L}$ denotes the deviation between the landmark configuration $\mathbf{q}_L$ and the starting configuration $\mathbf{q}_S$ and $\mathbf{R}_L$ is a positive definite diagonal weighting matrix. The variable P_l describes the path length starting from the parking configuration to the landmark, N_{SP} is the number of driving direction switching points along the path so far, and $r_l > 0$ and $r_{SP} > 0$ denote the corresponding weighting terms, respectively.

The cost function (8) is designed to rate each landmark based on its configuration with respect to the starting configuration and the path between parking configuration and the landmark. The weighting matrix $\mathbf{R}_L = \mathrm{diag}(\mathbf{R}, r_\theta)$ is used to consider the non-holonomic constraints of the car as explained in Section 3. The two remaining terms of the cost function intend to improve the quality of the path. Landmarks with a shorter path length and a lower number of direction switching points are chosen more frequently as initial configuration. Using this sorted list drastically reduces the computational costs.

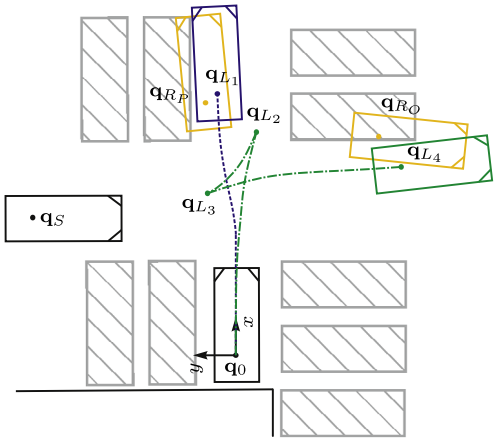


Fig. 8. Example for one path planning cycle starting from the initial configuration $\mathbf{q}_0$. Four landmarks $\mathbf{q}_{L_i}$, $i = 1, \ldots, 4$ are created in this example.

Fig. 8 illustrates one path planning cycle starting from the parking configuration $\mathbf{q}_0$ as initial configuration. Here, the landmark $\mathbf{q}_{L_1}$ is created for the parallel reference configuration $\mathbf{q}_{R_P}$ and three landmarks $\mathbf{q}_{L_2}$, $\mathbf{q}_{L_3}$, and $\mathbf{q}_{L_4}$ for the orthogonal reference configuration $\mathbf{q}_{R_O}$. Note that both reference configurations are not collision free but still four feasible landmarks are created.

Remark 1. The proposed method neither guarantees to find a path with minimal length nor with the minimal number of driving direction switching points. Nevertheless, extensive simulation studies show a good quality of the paths in the considered scenarios. However, some scenarios show an unnecessary large number of direction switching points. A method to eliminate such switching points could be to use the local path planner after the planning phase as optimisation tool. Planning between switching points along the path which are not connected with one draw decreases the overall number of switching points as well as the path length with low computing costs.

4.4. Algorithm

The path planning algorithm with the tree-based landmark creation is summarised in the flow chart in Fig. 9. The sorted list is initialised containing the parking configuration $\mathbf{q}_P$. The statement "*plan* $\mathbf{q}_A \rightarrow \mathbf{q}_B$" refers to the calculation of a local path from configuration $\mathbf{q}_A$ to $\mathbf{q}_B$ by the path planner of Algorithm 1. The local planner returns a path, a success indicator as well as the driving direction switching points $\mathbf{q}_{SP_i}$, $i = 1, \ldots, N_{L_p}$ including the final configuration of the path. The overall number of switching points and final configurations for one planning cycle is denoted by N_L.

Remark 2. The phases of the local path planner are initialised depending on the initial configuration $\mathbf{q}_0$. If the initial configuration $\mathbf{q}_0 = \mathbf{q}_P$, the local path planner is initialised in phase B. For all other initial configurations, the planning starts directly in phase A.

5. Simulation studies

To verify the path finding capability of the parking algorithm and the feasibility of the obtained paths, several simulation scenarios are investigated. The dimensions of the car are chosen similar to a mid-size commercial vehicle. The parameters are given in Table 1, whereby l_c represents the length and w_c the width of the car. Thus, the maximum control input u_l is given by $u_{l_{\max}} = \frac{\tan(\delta_{\max})}{l} = 0.2\ \mathrm{m}^{-1}$. The step length is limited to $\eta \in [0.05\ \mathrm{m}, 0.2\ \mathrm{m}]$. Note that the maximum step length should

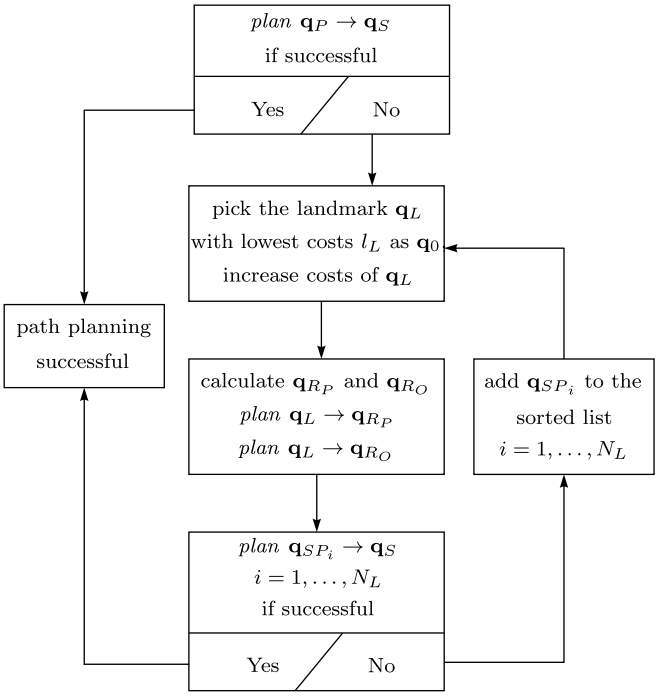


Fig. 9. Global path planning algorithm.

Table 1
Parameters of the car.

l_c	w_c	L	$\delta_{\max}$	$\zeta_{\max}$
5 m	1.9 m	2.9 m	30°	20°/s

not be too large as collision checks only occur at the start and end point of an optimisation step.

The environment is modelled in form of polygons where non-convex polygons are subdivided into multiple convex polygons. Collision checking is performed using the Minkowski sum [30]. The static optimisation problem is solved using the SQP-algorithm of the numeric software library NLopt [31,32]. All simulations are carried out in MATLAB on an Intel Core i7 3.4 GHz machine, where only one core was used for the calculations.

The optimisation parameters for both phases in all parking scenarios and the parameters for the tree-based extension are shown in Table 2. The parameters can be used to tune the quality of the final path. Higher values of r_u lead to a smoother steering angle at the cost of more driving direction switching points. The parameters r_l and r_{SP} can be used to obtain paths with shorter length and less direction changes, respectively, at costs of higher computing time. Higher values of Δx lead to longer paths in each planning cycle which is mainly useful for long, straight segments. The listed set of parameters focuses on a real-time capable solution while still remaining a good path quality.

All scenarios are compared with three path planning algorithms specialised on cars or car-like robots in cluttered environments. First, [33] is chosen as a typical representative of a roadmap planner. Second, [7] serves as a comparison method where a holonomic path is planned on a discrete grid. The non-holonomic path is obtained by connecting the grid cells of the holonomic path using Reeds and Shepp's curves. Finally, the Hybrid A* algorithm [18] which scored second place at the 2007 DARPA Urban Challenge is applied to the scenarios. Here, a tree with constant inputs is created which is guided by a heuristics based on a holonomic path and Reeds and Shepp's curves. The resolution and the number of trials of all methods are chosen that the average

Table 2
Parameters for path planning.

	Phase A	Phase B
r_θ	3.5	1
$\mathbf{R}$	diag(25, 1)	$\mathbf{0}$
r_u	0.1	
k	1.5	
r_l	10	
r_{SP}	50	
k_L	1.5	
Δx	6	

Table 3
Parking spot dimensions.

Scenario	a (m)	b (m)	d_S (m)	α (°)
Parallel	5.4	2.3	6.4	90
Garage	2.6	5.5	6.4	90
Angle	2.5	6.4	6.4	60

calculation time is three times larger than the calculation time of the proposed optimisation based planner. The results of all algorithms are reported without any optimisation procedure of the final path.

First, the convergence behaviour for parking scenarios at a street is demonstrated. The dimension of the parking spot for all scenarios is listed in Table 3 with the notation according to Fig. 1. The width of the street $d_S = 6.4$ m approximately corresponds to a standard street with two lanes. Note that in the parallel parking scenario the length of the parking spot is only 40 cm longer than the car. Fig. 10 depicts the planned paths of the proposed planner where the dashed lines symbolise the contour of the car at the parking and the starting configuration and the solid lines at the borders are the boundaries which must not be violated. The depicted parallel parking scenario requires five driving direction changes inside the parking spot to manoeuvre out of it. In the garage backwards and angle parking scenario two direction changes in phase A are used. All paths are planned with a calculation time less than 2 ms. The other path planners yield feasible paths for the garage and angle parking scenario but all fail for the parallel parking spot with the chosen resolution. Because of the little space in the parallel parking spot the resolution has to be very low to obtain a feasible path yielding no real-time capable solution.

Next, the convergence for typical parking manoeuvres is investigated. For this, several simulation studies in three different scenarios, namely a parking lot (PL), a parking deck (PD) and a narrow garage (NG) are demonstrated. The maximum number of driving direction switching points for each local path planning is set to $c_M = 3$. As representative for a cluttered environment, a parking lot (PL) is investigated first. At the parking lot, three, four and five 2 m long quadratic obstacles are placed randomly. This scenario was carried out in form of a Monte-Carlo simulation with 10^3 runs with a random starting configuration and randomly placed obstacles in each run. One representative for each, three, four and five obstacles, is presented in Fig. 11 where the path of the proposed planner (OBP planner) is drawn as solid line and the path obtained by the Hybrid A* algorithm as dashed line. The average calculation time of the proposed planner amounts to $t_{c,avg} = 20$–50 ms depending on the number of obstacles. The results of all four path planners are reported in Table 4 where the success rate $R_{\%}$, the average length of the path P_l and the average number of driving direction switching points N_{SP} are listed. The path planners are abbreviated by optimisation based path planner (OBP planner) for the proposed method, Hybrid A* for [18], Roadmap for [33] and Holonomic for [7]. All path planners manage to find a path in the majority of the scenarios where the proposed

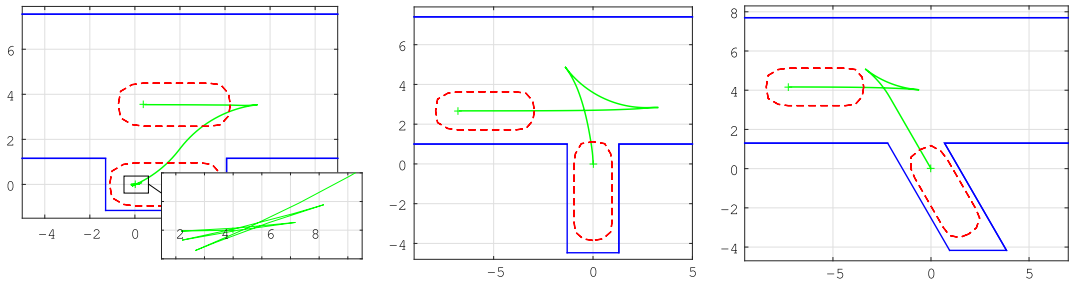


Fig. 10. Planned path for parallel, garage and angle parking; all quantities in m.

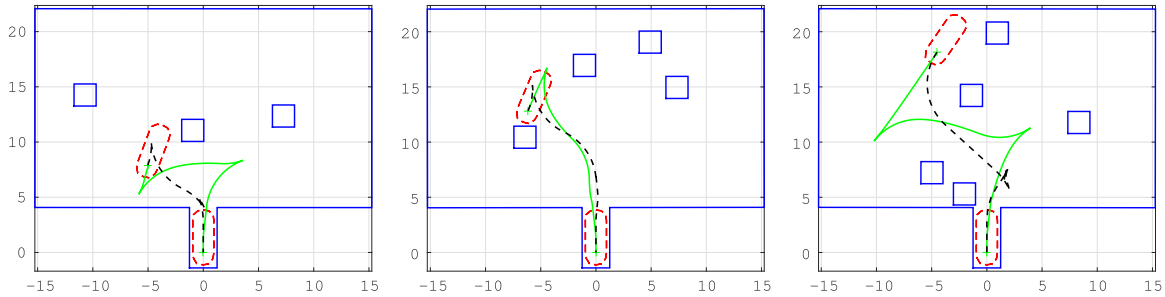


Fig. 11. Planned paths for a parking lot (PL) with randomly placed obstacles for the optimisation based planner (solid line) and the Hybrid A* algorithm (dashed line); all quantities in m.

Table 4
Simulation results of all four path planners.

Scenario	OBP Planner			Hybrid A*			Roadmap			Holonomic		
	$R_{\%}$	P_l (m)	N_{SP}	$R_{\%}$	P_l (m)	N_{SP}	$R_{\%}$	P_l (m)	N_{SP}	$R_{\%}$	P_l (m)	N_{SP}
PL 3 obstacles	94%	30.2	2	88.6%	17.8	1.3	84.2%	47.7	16	76.4%	28.3	11.2
PL 4 obstacles	91.6%	29.8	2.1	84%	17.7	1.3	83.3%	49.4	17	69.1%	28.8	11.6
PL 5 obstacles	89.4%	29	2.4	80.8%	17.5	1.4	78.2%	48	16.6	67.1%	28.4	11.1
PL 1 obstacle	100%	31.1	2.5	68.7%	19	1.8	88.5%	58.3	22	80.1%	30.7	13.4
PD backward	100%	25.9	3.3	0%	/	/	91.8%	66.4	36	100%	31.7	9
PD forward	99.2%	37.3	6.8	100%	17.4	2	78.4%	65.9	34.9	0%	/	/
NG backward	100%	24.8	3.9	100%	15.8	1	24.2%	44.2	22.9	0%	/	/
NG forward	100%	22.5	2	100%	11.5	2	90.0%	44.2	22.9	0%	/	/

planner exhibits the best convergence properties. Mainly with increasing number of obstacles, i.e., narrower environment, the optimisation based planner is able to outperform the other path planners with respect to convergence. The last method is able to find a holonomic path in nearly all scenarios but cannot connect this path using Reeds and Shepps curves in many scenarios because of the narrow environment. A path deformation method could increase its convergence. Note that there may not exist a feasible path due to the randomly placed obstacles. However, this does not influence the comparative result as all planners are tested on the same environment. The Hybrid A* algorithm yields in average shorter path with less driving direction changes because of the use of Reeds and Shepps curves. Due to the instantaneous change of the steering angle between the maximum angles, a path optimisation is deemed necessary which may increase the path length. The roadmap and holonomic planners have a much higher number of driving direction changes which have to be eliminated for real application. In general, the path quality of the proposed planner is sufficiently good to be applied to a real car.

To simulate a complex scenario, a rectangular obstacle is placed in front of the parking spot, see Fig. 12. Here, leaving the parking spot in one draw is not possible. The starting configuration is again chosen randomly and simulated over 10^3 runs. The results of all runs are also summarised in Table 4 (PL 1 obstacle). Here, in about 30% of the runs the Hybrid A* fails to find a feasible path. The roadmap and holonomic methods yield similar convergence properties as before but have a high number of driving direction changes. Two representative paths of the proposed planner (solid line) and the Hybrid A* (dashed line) are illustrated in Fig. 12. In the right subfigure, a configuration where the Hybrid A* algorithm fails to find a feasible path is depicted.

Next, two parking scenarios in a garage are investigated. The scenarios are depicted in Fig. 13, where the first one corresponds to a parking deck (PD), e.g., in a shopping mall where the car starts on a lane around the corner, and the second one illustrates a small narrow garage (NG), e.g., a personal basement garage in an apartment house. For the small garage, forwards and backwards parking is depicted. To account for the random elements of the path planners, the scenarios are carried out in 10^3 runs. The results are again summarised in Table 4. The proposed planner is the only planner to reliably find a path in (nearly) all scenarios with an average calculation time $t_{c,avg} \approx 20$ ms, which proves the real-time capability of the proposed approach. Again, no path with minimum length is found but the paths are suitable to be applied to a real car. The number of driving direction changes is sufficiently small. However, to decrease the number of direction changes and the path length, a post-processing optimisation could be employed.

Remark 3. The simple path optimisation scheme using the local path planner as mentioned in Remark 1 reduces the average number of switching points approximately by 0.2.

Finally, the obtained paths are tested for feasibility. Therefore, the dynamic model (1) is simulated in MatLab/Simulink with a

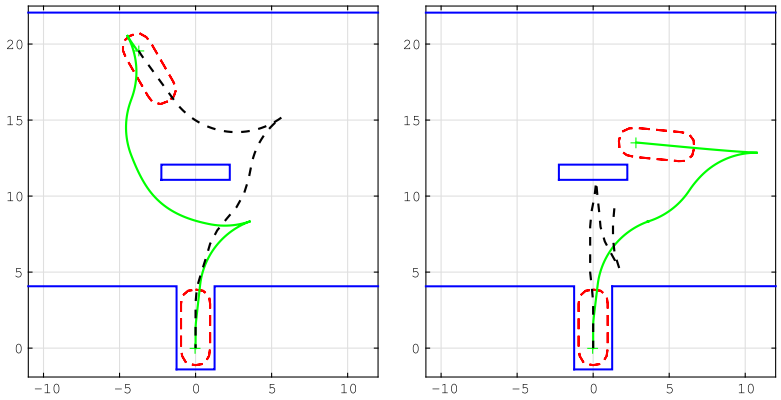


Fig. 12. Planned path for a parking lot with the optimisation based path planner (solid line) and the Hybrid A* path planner (dashed line); all quantities in m.

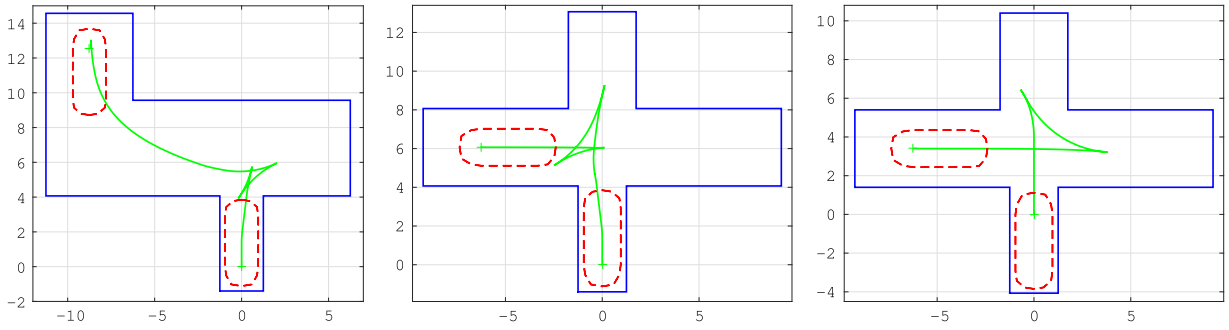
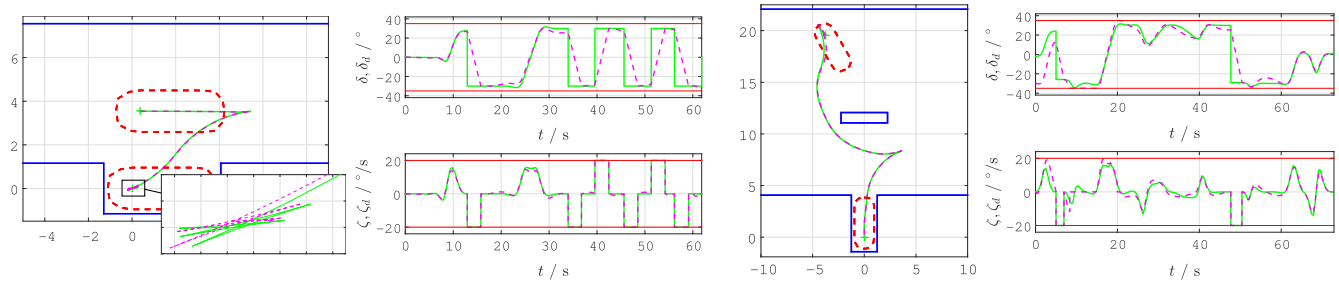


Fig. 13. Planned paths for a parking deck (PD) and a narrow garage (NG); all quantities in m.



(a) Parallel parking spot.

(b) Parking lot.

Fig. 14. Executed trajectory on the dynamic model.

time-parametrisation based on the flatness property of the system and a Lyapunov-based feedback control law as presented in [34]. For this, the parallel parking scenario and the scenario with one obstacle at the parking lot are exemplary chosen. Both scenarios are depicted in Fig. 14 where the planned (solid line) and executed (dashed line) paths are illustrated. Additionally, the planned and executed steering angle δ_d and δ and the planned and executed steering angle speed ζ_d and ζ are shown. The horizontal lines illustrate the maximum steering angle $\delta_{\max}$ and maximum steering angle speed $\zeta_{\max}$. In both scenarios, the steering angle is sufficiently smooth to execute the path without violating the maximum steering angle speed $\zeta_{\max}$. The error at the parking configuration is smaller than 3 cm and 3° for x, y and θ, respectively. All other planned paths exhibit similar behaviour but are not shown due to space restrictions.

6. Conclusions

In this paper, a fast optimisation based path planning algorithm for car parking in narrow environments is proposed. The path is discretised by a Runge–Kutta discretisation and calculated by recurrently solving a local static optimisation problem. The weighting terms for the optimisation are determined by dividing the path planning into two phases: one for the parking itself and one for driving to the parking spot. The choice of the switching points between these two phases and for direction changes is based on heuristic rules mimicking the behaviour of a human driver. To ensure good convergence properties in cluttered environment, a tree of landmarks is calculated using a reference configuration with very few requirements. The landmarks are implicitly created by the local path planner leading to a good coverage of the free configuration space also close to obstacles.

Simulations for different scenarios show the feasibility of the proposed algorithm. Without any modifications, parallel, garage and angle parking problems can be handled in a straightforward way. The path planning can be carried out within a few milliseconds even in narrow environments. It can therefore also be implemented in model predictive control schemes which systematically account for moving obstacles. For longer driving distances at a parking deck or on a parking lot, feasible paths can be obtained based on the derived tree of landmarks. The benefits of the proposed approach compared to the state of the

art, in particular the Hybrid A* path planner [18], the roadmap planner according to [33] and the holonomic planner due to [7], are demonstrated by Monte Carlo simulations of narrow parking deck and parking lot scenarios. For all scenarios, the proposed planner is able to compete or outperform the compared path planners. The slightly longer path length is in general no problem for real applications as very short path lengths between driving direction changes are often an undesired behaviour. However, a post-processing path optimisation has the potential to eliminate unnecessary direction changes and further shorten the path length.

Extensive simulation studies prove that the algorithm is able to provide feasible paths for various scenarios within a considerably low computing time. However, the local nature of the underlying optimisation problem together with the heuristics in choosing the switching points leaves open several questions concerning a rigorous mathematical proof of the convergence. Simulation studies on the dynamic model of the car show the feasibility of the paths. Nevertheless, future work will address implementing the path planning algorithm on a real car or a car-like robot to verify the behaviour under measurement noise, sensor uncertainties and dynamic obstacles.

References

[1] P. Bessiere, J.-M. Ahuactzin, E.-G. Talbi, E. Mazer, The "Ariadne's clew"; algorithm: global planning with local methods, in: Proc. on Int. Conf. on Intelligent Robots and Systems, Vol. 2, Yokohama, Japan, July 1993, pp. 1373–1380.

[2] E. Mazer, J.M. Ahuactzin, P. Bessière, The Ariadne's clew algorithm, J. Artificial Intelligence Res. 9 (1998) 295–316.

[3] L. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE J. Robot. Autom. 12 (4) (1996) 566–580.

[4] S. Chakravorty, S. Kumar, Generalized sampling-based motion planners, IEEE Trans. Syst. Man Cybern. B 41 (3) (2011) 855–866.

[5] J.J. Kuffner, S.M. LaValle, RRT-connect: An efficient approach to single-query path planning, in: Proc. IEEE Int. Conf. on Robotics and Automation, Vol. 2, San Francisco, CA, USA, April 2000, pp. 995–1001.

[6] D. Hsu, R. Kindel, J.-C. Latombe, S. Rock, Randomized kinodynamic motion planning with moving obstacles, in: Proc. 4th Int. Workshop on the Algorithmic Foundations of Robotics, Hanover, NH, USA, March 2000, pp. 247–264.

[7] J.-C. Latombe, A fast path planner for a car-like indoor mobile robot, in: Proc. National Conference on Artificial Intelligence, Vol. 2, Anahein, CA, USA, July 1991, pp. 659–665.

[8] L. Dubins, On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents, Amer. J. Math. 79 (3) (1957) 497–516.

[9] J.A. Reeds, L.A. Shepp, Optimal paths for a car that goes both forwards and backwards, Pacific J. Math. 145 (2) (1990) 367–393.

[10] M.F. Hsieh, U. Özgüner, A parking algorithm for an autonomous vehicle, in: Proc. IEEE Intelligent Vehicles Symposium, Eindhoven, Netherlands, June 2008, pp. 1155–1160.

[11] K. Lee, D. Kim, W. Chung, H.W. Chang, P. Yoon, Car parking control using a trajectory tracking controller, in: Proc. Int. Joint Conf. SICE-ICASE, Busan, Korea, October 2006, pp. 2058–2063.

[12] S.K. Lee, S. Lee, C. Nam, N.L. Doh, Local path planning scheme for car-like vehicle's shortest turning motion using geometric analysis, in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Taipei, Taiwan, October 2010, pp. 4761–4768.

[13] D. Kim, W. Chung, S. Park, Practical motion planning for car-parking control in narrow environment, IET Control Theory Appl. 4 (1) (2010) 129–139.

[14] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, K.-i. Machida, Curvature continuous path generation for autonomous vehicle using B-spline curves, Comput.-Aided Des. 42 (4) (2010) 350–359.

[15] F. Gómez-Bravo, F. Cuesta, A. Ollera, A. Viguria, Continuous curvature path generation based on β-spline curves for parking manoeuvres, Robot. Auton. Syst. 56 (4) (2008) 360–372.

[16] J. Pérez, J. Godoy, J. Vallagrá, E. Onieva, Trajectory generator for autonomous vehicles in urban environments, in: Proc. IEEE Int. Conf. on Robotics and Automation, Karlsruhe, Germany, May 2013, pp. 409–414.

[17] S.S. Ge, X.-C. Lai, A.A. Mamun, Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints, Robot. Auton. Syst. 55 (7) (2007) 513–526.

[18] D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, Path planning for autonomous vehicles in unknown semi-structured environments, Int. J. Robot. Res. 29 (5) (2010) 485–501.

[19] J.-P. Laumond, P. Jacobs, M. Taix, R. Murray, A motion planner for nonholonomic mobile robots, IEEE J. Robot. Autom. 10 (5) (1994) 577–593.

[20] J. Laumond, S. Sekhavat, F. Lamiraux, Guidelines in nonholonomic motion planning for mobile robots, in: J.-P. Laumond (Ed.), Robot Motion Planning and Control, Springer, Berlin, 1998, pp. 1–54.

[21] S. Sekhavat, J.-P. Laumond, Topological property for collision-free nonholonomic motion planning: the case of sinusoidal inputs for chained form systems, IEEE J. Robot. Autom. 14 (5) (1998) 671–680.

[22] N. Ghita, M. Kloetzer, Trajectory planning for a car-like robot by environment abstraction, Robot. Auton. Syst. 60 (4) (2012) 609–619.

[23] B. Müller, J. Deutscher, S. Grodde, Continuous curvature trajectory design and feedforward control for parking a car, IEEE Trans. Control Syst. Technol. 15 (3) (2007) 541–553.

[24] T. Fraichard, A. Scheuer, From Reeds and Shepp's to continuous-curvature paths, IEEE Trans. Robot. 20 (6) (2004) 1025–1035.

[25] B. Mirtich, J. Canny, Using skeletons for nonholonomic path planning among obstacles, in: Proc. IEEE Int. Conf. on Robotics and Automation, Vol. 3, Nice, France, May 1992, pp. 2533–2540.

[26] K. Kondak, G. Hommel, Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms, in: Proc. Int. Conf. on Robotics & Automation, Vol. 3, Seoul, Korea, May 2001, pp. 2698–2703.

[27] P. Zips, M. Böck, A. Kugi, A fast motion planning algorithm for car parking based on static optimization, in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Tokyo, Japan, November 2013, pp. 2392–2397.

[28] K. Kant, S.W. Zucker, Towards efficient trajectory planning: The path-velocity decomposition, Int. J. Robot. Res. 5 (3) (1986) 72–89.

[29] H. Liu, W. Liu, L. Latecki, Convex shape decomposition, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, San Francisco, CA, USA, June 2010, pp. 97–104.

[30] T. Lozano-Pérez, Spatial planning: A configuration space approach, IEEE Trans. Comput. C-32 (2) (1983) 108–120.

[31] S.G. Johnson, The NLopt nonlinear-optimisation package, March 2015. [Online]. Available: http://ab-initio.mit.edu/nlopt.

[32] D. Kraft, A software package for sequential quadratic programming, Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen, Germany, 1988.

[33] J. Qu, S.M. LaValle, Nonholonomic mobile robot motion planning—motion strategy project. March 2015, 1999. [accessed] April 2015. [Online]. Available: http://msl.cs.uiuc.edu/~lavalle/cs576_1999/projects/junqu/.

[34] P. Zips, M. Böck, A. Kugi, Fast optimization based motion planning and path-tracking control for car parking, in: Proc. IFAC Symposium on Nonlinear Control Systems, Toulouse, France, September 2013, pp. 86–91.

Patrik Zips received the B.Sc. and Dipl.-Ing. degrees in electrical engineering and the Ph.D. (Dr.techn.) degree in control engineering from Vienna University of Technology, Austria, in 2010, 2012 and 2015 respectively.

Since 2012, he works as a research assistant at the Automation and Control Institute at Vienna University of Technology. His research interests include optimisation based path planning and differential geometric as well as optimisation based methods for nonlinear control.

Martin Böck received the Dipl.-Ing. degree in mechatronics from Johannes Kepler University, Linz, Austria, and the Ph.D. (Dr. techn.) degree in control engineering from Vienna University of Technology, Austria, in 2010 and 2016, respectively.

Since 2010, he works as a research assistant at the Automation and Control Institute at Vienna University of Technology. His research interests include distributed optimisation, optimisation based control methods, model predictive control and differential geometric methods for nonlinear control.

Andreas Kugi received the Dipl.-Ing. degree in electrical engineering from Graz University of Technology, Austria, and the Ph.D. (Dr. techn.) degree in control engineering from Johannes Kepler University (JKU), Linz, Austria, in 1992 and 1995, respectively. He received his "Habilitation" degree in the field of automatic control and control theory from JKU in 2000.

From 1995 to 2000 he worked as an assistant professor and from 2000 to 2002 as an associate professor at JKU. In 2002, he was appointed full professor at Saarland University, Saarbrücken, Germany, where he held the Chair of System Theory and Automatic Control until May 2007. Since June 2007 he is a full professor for complex dynamical systems and head of the Automation and Control Institute at Vienna University of Technology, Austria. His research interests include the physics-based modelling and control of (nonlinear) mechatronic systems, differential geometric and algebraic methods for nonlinear control, model predictive control and the controller design for infinite-dimensional systems. He is full member of the Austrian Academy of Sciences and Editor-in-Chief of the *Control Engineering Practice*.