2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

A Fast Motion Planning Algorithm for Car Parking Based on Static Optimization

Patrik Zips, Martin Böck, Andreas Kugi

Abstract— **This paper presents a fast optimization based algorithm for car parking. The challenge arises from the non-holonomic characteristics of the car and the close distance to the obstacles. The presented approach utilizes the Minkowski sum to account for obstacle avoidance. The geometric path planning problem is decoupled from the kinematic problem and discretized with respect to the path parameter by means of a Runge-Kutta discretization. For the discrete path segments, an optimization problem is formulated to calculate the path independent of the parking scenario. This static optimization problem can be solved numerically in a very efficient way. The performance of the algorithm is evaluated in several simulation scenarios.**

I. Introduction

One topic of recent research in automobile industry is autonomous driving, which is especially challenging in urban environments, where narrow corridors, tight turns and unpredictable moving obstacles like other cars have to be handled. A special topic within this area of research is automatic parking control, which is not only useful for autonomous driving vehicles, but also in conventional cars as parking assistance system.

In the last decades, a number of different approaches have been developed to tackle this problem. Many of these concepts rely on the ideas of Reeds and Shepp that the shortest path for a car which goes forwards and backwards in obstacle free environment is composed of minimum curvature arcs and lines due its non-holonomic constraints [1]. By stringing together such lines and minimum curvature arcs a path into a parking spot can be constructed, see, e.g., [2], [3]. In [4] every possible combination of a predefined number of line-arc-line cycles is computed and one of all calculated paths is chosen on the basis of a cost criterion.

Instead of calculating all possible paths and choosing one of them, the optimal path defined by a cost criterion can be directly obtained by solving an optimal control problem subject to the dynamic constraints of the car [5], [6]. Hereby not only the geometric path can be calculated, but also control inputs like acceleration and steering angle change are provided at the drawback of higher computational costs. The main challenge within this approach is to properly incorporate the obstacles into the mathematical formulation.

One possibility to account for the obstacles is given by the so called navigation function, see, e.g., [7], [8]. This function describes the environment by a potential field, where the global minimum of this field corresponds to the target point and obstacles are modeled by a high potential. This function is then added to the cost function. As the obstacles are not included as hard constraints, there is in general no guarantee that the path is collision-free.

Another approach is to discretize the obstacles to obtain boundaries composed of a finite number of points [9]. Every point has a potential field value which depends on the vehicle position. As long as no collision occurs these values are set to zero, otherwise a value greater than zero is assigned. The values of all points are summed up in one inequality constraint which has to be lower or equal zero. Only if no point collides with the vehicle this inequality holds.

A different approach to handle the non-holonomic nature of the car was proposed by Laumond et al. Thereby, first a holonomic path is computed for a given environment and then this path is followed under consideration of the non-holonomic constraints [10]. Due to the fact that a car is small-time-controllable [5] this is always possible, but often results in highly maneuvering paths in particular in narrow environments. There are different methods to follow a holonomic path. In [11] the differential equations of the car are transformed into a chained-form system and sinusoidal inputs are applied. In [12] a local continuous curvature planner using clothoids in combination with a shortest feasible path metric is used to obtain the non-holonomic path.

In this paper, we propose a new optimization based algorithm focused on car-parking problems with low computational costs for real-time applications. We consider three different parking scenarios, which, together with the kinematic car model, are presented in Section II. A method for planning the path by solving a static optimization problem is presented in Section III. Simulation studies for different parking scenarios showing the practical feasibility of the algorithm are carried out in Section IV.

II. Problem Statement

In this paper, we deal with three common parking scenarios, namely parallel, garage and angle parking. These scenarios and the corresponding notations are shown in Fig. 1, where the shaded polygons represent obstacles like other cars. The lines to the left and right side of each scenario are boundaries, which shall not be violated like, e.g., the kerbstone or the lane separator of the street.

The algorithm has to find a feasible path from a given starting configuration to a parking position. The path shall have a reasonable length and must not collide with any obstacle or exceed a boundary. Moreover, the same motion planner shall be able to plan a path for all three scenarios.

The authors are with the Automation and Control Institute, Vienna University of Technology, 1040 Vienna, Austria (e-mail: {zips,boeck,kugi}@acin.tuwien.ac.at).

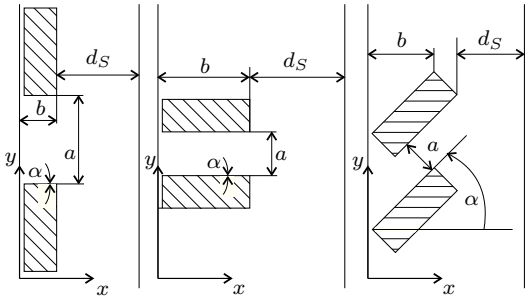

Fig. 1. Notation for parallel, garage and angle parking.

A. Obstacles

Without loss of generality we concentrate on convex polygonal obstacles. Every non-convex polygon can be subdivided into multiple convex polygons, which is usually referred to as convex decomposition [13]. Non-polygonal obstacles can always be included in a slightly larger polygon.

B. System Dynamics

In the following, the kinematic model with Ackerman steering, as shown in Fig. 2, serves as basis for the mathematical description of the car behavior. Thereby the tire slip angle is neglected, which is justified by the low velocity during parking maneuvers. Hence the car can be described by one front and one rear wheel. The motion of the car is characterised by the coordinates (x, y) of a reference point P_R, which is located at the center of the rear axle, as well as the orientation θ of the longitudinal axis of the car. The non-holonomic kinematic differential equations in these generalised coordinates $\mathbf{q} = [x, y, \theta]^T$ read as

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

Here $\mathbf{u}_A = [v, \delta]^T$ describes the control input with the velocity v and the steering angle δ, and the parameter L denotes the wheelbase. The geometric path planning can be decoupled from the kinematic velocity planning by Path-Velocity-Decomposition [14]. 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 (1) can be rewritten as

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

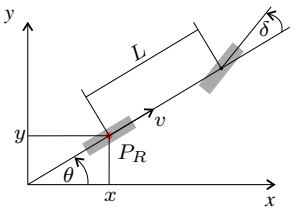

Fig. 2. Kinematic model of the car.

where $(\cdot)'$ denotes the derivative with respect to s and $u_l = \frac{\tan(\delta)}{L}$ is the new control input. The Path-Velocity-Decomposition simplifies the problem, as the time is neglected within the path planning, resulting in lower computational costs.

III. Motion Planning

The parking task can be formulated as an optimal control problem (OCP) of the form

$$\min_{u_l(s), D(s)} \; J(u_l(s), D(s)) = \int_{s_0}^{s_1} l(\mathbf{q}, u_l, D)\mathrm{d}s \tag{3a}$$

$$\text{s.t. } \mathbf{q}' = \mathbf{f}_s(\mathbf{q}, u_l, D), \; \mathbf{q}(s_0) = \mathbf{q}_S, \mathbf{q}(s_1) = \mathbf{q}_P \tag{3b}$$

$$\mathbf{h}_P(\mathbf{q}) \leq \mathbf{0} \tag{3c}$$

$$|u_l| \leq u_{l_{max}}, \tag{3d}$$

with an appropriate cost function $l(\mathbf{q}, u_l, D)$. Note that (3b) corresponds to the kinematic model (2), where $\mathbf{q}_S$ and $\mathbf{q}_P$ denote the generalized coordinates of the car related to the starting position and the desired parking position, respectively. Furthermore, the control input u_l is constrained by the maximum steering angle δ_{max} with $u_{l_{max}} = \frac{\tan(\delta_{max})}{L}$ and $D \in \{-1, 1\}$ describes the driving direction. To account for collision avoidance each polygon is transformed into an inequality constraint h_{P_j} by means of the Minkowski sum [15]. All polygon inequality constraints are combined in $\mathbf{h}_P(\mathbf{q}) = [h_{P_1}(\mathbf{q}), \ldots, h_{P_P}(\mathbf{q})]^T$.

In general, calculating the optimal solution of the mixed-integer optimal control problem (3) is computationally demanding. Especially the determination of $D(s)$, i.e. the direction switching points along the path, is numerically challenging. These facts make a real-time solution of the OCP (3) under reasonable effort impossible.

Therefore, we propose an alternative real-time capable motion planner, which relies on a constrained static optimization problem. To determine an appropriate cost function for the optimization problem, the parking algorithm is divided into two phases. Based on these phases the weighting terms for the cost function as well as two rules for direction changes are determined.

A. Principle of the algorithm

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 the parking spot. For this a suitable path between two predefined points must be found, mostly in an obstacle-free environment.

Phase B handles the parking maneuver itself, where typically small but precise position and orientation changes in narrow environments have to be achieved. The change between phase 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 from the parking position via the phase switching point to the starting position. Thus the first task is to find a suitable

path from the parking position to the phase switching point, from where it is ensured that the car is able to leave the parking spot.

This task is trivial for garage and angle parking as the car only has to drive straight ahead or backwards. For parallel parking it is more challenging, because the algorithm has to find a way in a narrow environment by switching between driving forwards and backwards. Therefore, we will at first concentrate on a suitable strategy for parallel parking. There are two possible methods for a car to get out of a narrow parallel parking spot. First it can use its small-time-controllability property to drive nearly sidewards by driving forwards and backwards with small steering efforts [5]. Second it can change its orientation in the parking spot far enough to be able to drive straight ahead out of the parking spot. As the latter method results in smoother paths and less maneuvering, we strive for imitating this behavior. Consequently in this phase, the path planner has to change the car orientation θ to a suitable value while avoiding collisions with obstacles.

The second task in reverse path planning is concerned with the path from the phase switching point to the starting position. Usually, this is straightforward and does not require more than one direction change.

For determining the path of each phase, the same constrained static optimization problem will be formulated, which only differs in the weighting of the cost function. Furthermore, a strategy for calculating the phase switching point and for direction changes will be proposed. The basic idea of the whole parking algorithm is based on mimicking the behavior of a human driver.

B. Discretization and static optimization problem

In a first step, the differential equation (2) is discretized with respect to the path parameter s by means of a Runge-Kutta discretization of second order

$$\mathbf{k}_{1_i} = \eta_i \mathbf{f}_s\left(\mathbf{q}_i, u_{l_i}\right) \tag{4a}$$

$$\mathbf{k}_{2_i} = \eta_i \mathbf{f}_s\left(\mathbf{q}_i + \frac{1}{2}\mathbf{k}_{1_i}, u_{l_i}\right) \tag{4b}$$

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \mathbf{k}_{2_i} + \mathcal{O}\left(\eta_i^3\right), \tag{4c}$$

with the step length $\eta_i \in [\eta_{min}, \eta_{max}]$. Note that the step length is constrained by a minimum and maximum value, η_{min} and η_{max}, respectively. Neglecting the error term in (4) this yields the difference equation

$$\begin{aligned}\mathbf{q}_{i+1} &= \begin{pmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + D\eta_i \cos\left(\theta_i + D\frac{\eta_i u_{l_i}}{2}\right) \\ y_i + D\eta_i \sin\left(\theta_i + D\frac{\eta_i u_{l_i}}{2}\right) \\ \theta_i + D\eta_i u_{l_i} \end{pmatrix} \\ &= \mathbf{f}(\mathbf{q}_i, \mathbf{u}_i, D),\end{aligned} \tag{5}$$

with $D \in \{-1, 1\}$ for negative and positive velocities, respectively. The new control input $\mathbf{u} = [u_l, \eta]^T$ consists of the steering input u_l and the step length η.

In contrast to (3), the optimization problem will not be formulated for the whole path at once but only for one incremental step. This procedure is then applied in a recursive manner. In each iteration it is decided whether a direction change of the car is deemed necessary and if the phase switching point is reached. The next subsection will detail how these switching points are chosen by the algorithm.

The constrained static optimization problem to be solved in each iteration takes the form

$$\min_{\mathbf{u}_i} \; l_{O_i}(\mathbf{q}_{i+1}) \tag{6a}$$

$$\text{s.t. } \mathbf{q}_{i+1} = \mathbf{f}\left(\mathbf{q}_i, \mathbf{u}_i, D\right) \tag{6b}$$

$$\mathbf{h}_P\left(\mathbf{q}_{i+1}\right) \leq \mathbf{0} \tag{6c}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_i \leq \mathbf{u}_{max}, \tag{6d}$$

with the cost function

$$l_{O_i}(\mathbf{q}_{i+1}) = r_\theta e_{\theta_{i+1}}^2 + \mathbf{e}_{P_{i+1}}^T \mathbf{R} \mathbf{e}_{P_{i+1}}. \tag{7}$$

Thereby, $\mathbf{e}_{P_i} = [x_i - x_S, y_i - y_S]^T$ denotes the distance of the car at iteration step i to the starting position[1] $[x_S, y_S]^T$ and $e_{\theta_i} = \theta_i - \theta_O$ refers to the difference between the car orientation and a predefined target angle θ_O. The parameter $r_\theta > 0$ and the positive semi-definite matrix $\mathbf{R}$ serve as weighting terms in the cost function. The constraint (6b) corresponds to (5), (6c) accounts for collision avoidance and the control input $\mathbf{u}_i$ is constrained by $\mathbf{u}_{min} = [-u_{l_{max}}, \eta_{min}]^T$ and $\mathbf{u}_{max} = [u_{l_{max}}, \eta_{max}]^T$.

In every iteration step the position and the orientation of the car with respect to the cost function (7) is improved. The cost function is designed to fulfill the previously defined tasks in phase A and phase B by assigning suitable values to the weighting terms r_θ and $\mathbf{R}$ as well as to the target angle θ_O.

This can be accomplished by examining in more detail the role of phase A and B in the parking algorithm. In phase B, the algorithm should provide a path such that the car is able to leave the parking spot. Therefore, the car orientation θ is most important whereas the weighting matrix $\mathbf{R}$ can be set to zero or to a considerably small value. 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 and increased, respectively. A target angle θ_O, which may be used for all scenarios, is the angle α between the boundary below the car and the x-axis as plotted in Fig. 1, i.e. in phase B $\theta_O = \alpha$.

As already mentioned in the previous subsection, in phase A the parking algorithm has to find a suitable path from the phase switching point to the starting position. Therefore, in phase A the target angle is set equal the starting angle $\theta_O = \theta_S$. The choice of the weighting parameters is important for finding the switching points for the necessary direction changes as will be described in the next subsection.

[1]Remember that the path is planned in reverse direction from the parking to the starting position.

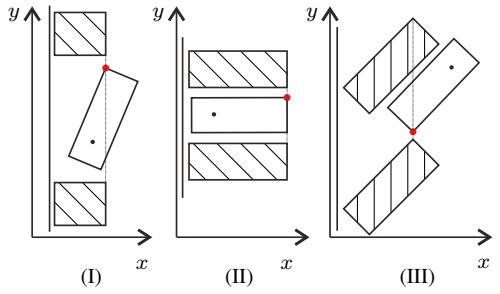

Fig. 3. Phase switching point for parallel (I), garage (II) and angle parking (III) for left parking spots.

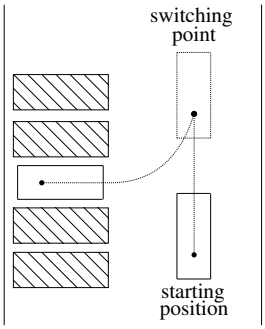

Fig. 4. Vehicle needs to switch direction while driving to the parking spot.

C. Switching points

Next we want to define suitable switching points to change between the phases and for direction changes. First we will present a condition for the phase switching point and afterwards two rules for direction changing.

1) Phase switching point: From the phase switching point the car shall be able to leave the parking spot and drive towards the starting position. As a condition we propose that the front corner of the car on the side of the parking spot has to pass the artificial connection line of the inner edge of the outer obstacle boundaries. This is illustrated in Fig. 3 for a left parallel, garage and angle parking scenario. In our notation shown in Fig. 1, this condition can be written as $x_{C_{FL}} \geq b$ for parking spots to the left of the car and $x_{C_{FR}} \leq b$ to the right of it. Thereby, $x_{C_{FL}}$ and $x_{C_{FR}}$ describe the x-position of the front left and front right corner of the car, respectively.

2) Switching points for direction change: Calculating optimal positions for direction switching points is a complex task [5], [16]. Therefore, we introduce two heuristic rules, which basically mimic the behavior of a human driver. Clearly this strategy does not yield optimum paths in the sense of (3) but has the advantage of considerably low computational costs.

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 η_{min}. This means the optimization problem (6) has no feasible solution subject to the constraints. At this point the direction is switched, i.e. D in (5) is set to $-D$, and the optimization problem is solved again. If this yields no solution the algorithm does not find a feasible path into the parking spot. The reason for this is either that the parking spot is too small or that the parking position is too close to the kerbstone. To avoid the latter case, the desired parking position can be placed sufficient far away from the kerbstone.

The second rule is primarily designed for phase A. Here switching points may be important for the car to reach the starting position. An example is shown in Fig. 4 for garage parking. The starting position cannot be reached from the parking position in one draw. The best way is to drive out of the parking spot to the left and change direction when the orientation angle θ is sufficient large. At this point the choice of the weighting terms for phase A gets evident. First the x-position and the orientation angle θ must be close enough to the starting position and afterward the y-position can be easily reached by driving straight for- or backwards.

In optimization terms, the cost function will decrease until the θ- and x-deviation to the starting position are sufficiently small and an increase of the y-deviation causes an increasing cost function. At this point $l_{O_i} > l_{O_{i-1}}$, the direction is switched and the optimization problem solved again.

The condition for the phase switching point as well as the second rule for direction changes are the reasons why the path is planned backwards. If the algorithm would plan forwards, coordinates for all switching points need to be defined in advance instead of these two simple conditions.

D. Parking algorithm

Algorithm 1 shows the complete parking algorithm for a parking spot on the left side, whereby r_{θ_A} and $\mathbf{R}_A$ denote the weighting terms for phase A and the superscript * refers to the optimal values. Thus $l^*_{O_i} = l_{O_i}(\mathbf{q}^*_{i+1})$ and $\mathbf{q}^*_{i+1} = f(\mathbf{q}_i, \mathbf{u}^*_i)$. The starting and parking positions $\mathbf{q}_S$ and $\mathbf{q}_P$ are assigned beforehand. The algorithm stops if the current position is in an ϵ neighborhood of the starting position. For a parking spot to the right the **if** condition in line 8 has to be changed to $x_{C_{FR},i} \leq b$.

Although it is a local planning algorithm, global convergence for specific geometric conditions can be shown. For the sake of conciseness we omit the detailed proof at this point but refer to forthcoming publications. Nevertheless, the simulation studies presented in the next section prove the feasibility for typical parking scenarios.

IV. Simulation studies

To verify the performance and path finding capability of the algorithm, several simulation scenarios are investigated. The dimensions of the car are chosen similar to a mid-sized commercial vehicle. The parameters are shown in Table I, whereby l_c represents the length and w_c the width of the car.

The maximum control input u_l is therefore given by $u_{l_{max}} = \frac{\tan(\delta_{max})}{L} = 0.37$. The step length is limited to $\eta \in [1 \cdot 10^{-3}\text{m}, 0.2\text{m}]$. The maximum step length should not exceed a certain value as collision checks only occur at the start and end point of an optimization step.

Algorithm 1 Parking algorithm

$\mathbf{q}_0 = \mathbf{q}_P$ {parking position}
$\theta_O = \alpha$ {target angle}
$r_\theta = 1, \mathbf{R} = \mathbf{0}$ {weighting terms}
$D = 1$ {direction = forward}
$\mathbf{u}_0 = [0, \eta_S]^T$ {initial guess}
$l_{O_0} = \infty, i = 0$
repeat
  if $x_{C_{FL},i} \geq b$ **then**
    $\theta_O = \theta_S$, $r_\theta = r_{\theta_A}, \mathbf{R} = \mathbf{R}_A$
  end if
  $\mathbf{u}_i^* = \arg\min_{\mathbf{u}_i} \; l_{O_i}(\mathbf{q}_{i+1})$
    s.t. $\mathbf{q}_{i+1} = \mathbf{f}(\mathbf{q}_i, \mathbf{u}_i, D)$
      $\mathbf{h}_P(\mathbf{q}_{i+1}) \leq 0$
      $\mathbf{u}_{min} \leq \mathbf{u}_i \leq \mathbf{u}_{max}$
  if $l^*_{O_i} > l^*_{O_{i-1}}$ **or** $\mathbf{h}_P(\mathbf{q}^*_{i+1}) > 0$ **then**
    $D = -D$
  else
    $\mathbf{q}_{i+1} \leftarrow \mathbf{f}(\mathbf{q}_i, \mathbf{u}_i^*, D)$
    $i \leftarrow i + 1$
  end if
until $||\mathbf{q} - \mathbf{q}_S|| \leq \epsilon$

TABLE I
PARAMETERS OF THE CAR.

l_c	w_c	L	δ_{max}
4.7m	1.8m	2.7m	45°

The optimization parameters for both phases and all three scenarios are shown in Table II. To improve convergence behavior for garage and angle parking the first element of $\mathbf{R}$ in phase B is set to $R_{1,1} = 0.1$ although it also works with $\mathbf{R} = \mathbf{0}$.

For obstacles at the border of the configuration space $\mathcal{C}$ only the boundaries in the interior of $\mathcal{C}$ are considered. A convex decomposition is performed by checking the angles between two adjacent boundaries inside the obstacle. If the angle is convex, these boundaries are summarized to one polygon. If a nonconvex angle appears, a new polygon is added to the list. An example for a parallel parking spot is show in Fig. 5, whereby Roman numerals are used to name the polygons. This method does not yield closed polygons for all boundaries but nevertheless suitable inequality constraints.

The static optimization problem is solved using the SQP-algorithm of the numeric software package SNOPT [17]. All simulations are carried out in MATLAB on an Intel Core i7 3.4GHz machine.

TABLE II
OPTIMIZATION PARAMETERS FOR PHASE A AND B.

	Phase A	Phase B
r_θ	4	1
θ_O	θ_S	α
$\mathbf{R}$	$\begin{pmatrix} 30 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0 \end{pmatrix}$

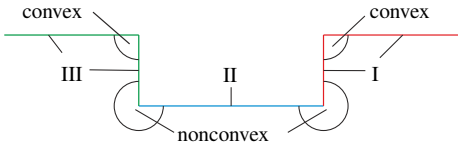

Fig. 5. Convex decomposition for a parallel parking spot.

TABLE III
PARKING SPOT DIMENSION AND SIMULATION RESULTS.

scenario	a/m	b/m	d_S/m	α	t_c/ms	N_{iter}
parallel 1	6	2.2	4	0°	6	34
parallel 2	5.1	2.2	4	0°	9	46
garage	2.3	5	7	0°	10	72
angle	2.26	4.7	5	45°	8	54

The left part of all illustrated simulation figures shows the trajectory of the reference point P_R and the right part the path of the car with its boundary drawn. All simulation scenarios with the corresponding parking spot dimensions, the calculation time t_c and the number of iterations N_{iter} of the algorithm are summarized in Table III.

We start with demonstrating two scenarios for parallel parking. The first one is a rather large parking spot, in which the car can drive into without changing direction. The second scenario is concerned with a very small parking spot to demonstrate the capability of the algorithm. We choose the parking spot just 40cm larger than the car length, which is less than 7cm larger than the diagonal of the car.

In both scenarios the algorithm finds a feasible path. The first scenario just needs a computation time of $t_c = 6$ms and is depicted on the left hand side of Fig. 6. The second scenario is shown on the right hand side of Fig. 6. The algorithm has to switch directions of the car multiple times but still manages to find a feasible trajectory within $t_c = 7$ms. The maneuvering in the parking spot is shown in more detail in the enlarged subfigure. This scenario shows the effectiveness of the proposed algorithm.

Next we consider a garage parking scenario. As already explained before, the parking itself in this case is rather trivial. To demonstrate the behavior of the algorithm in the obstacle free space, we choose a starting position, from where the car cannot get into the parking spot in the desired orientation without changing the direction. The calculated path is obtained in $t_c = 10$ms calculation time and shown on the left hand side of Fig. 7. By increasing r_θ for this phase the steering effort could be reduced with the drawback of a longer path. In this context it should be noted that only the starting and the parking position are defined, but no point for direction switching.

Finally, we show an angle parking scenario. Although this constitutes an easy parking scenario, it demonstrates the generality of the proposed algorithm. The definition of the target angle $\theta_O = \alpha$ ensures that a feasible path for driving directly into the parking spot is found, as shown on the right hand side of Fig. 7. Again the calculation time with $t_c = 8$ms

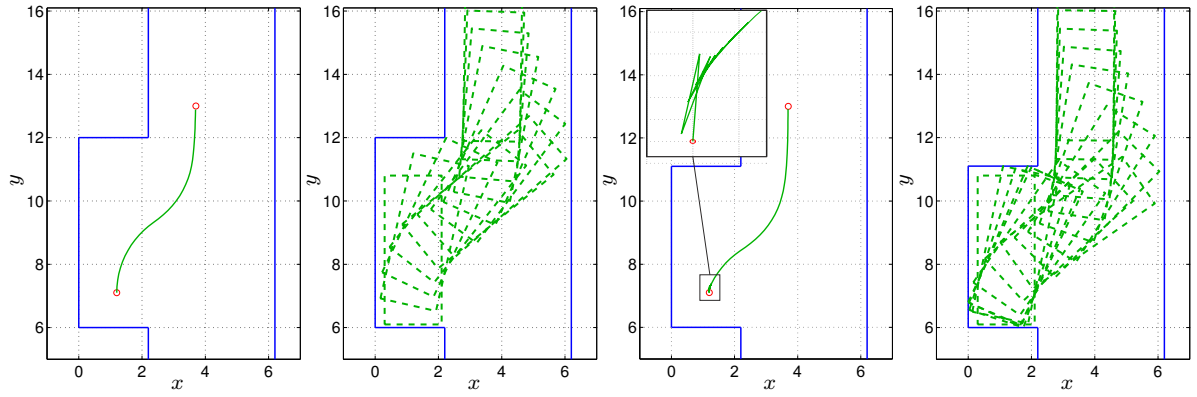

Fig. 6. Parallel parking scenarios.

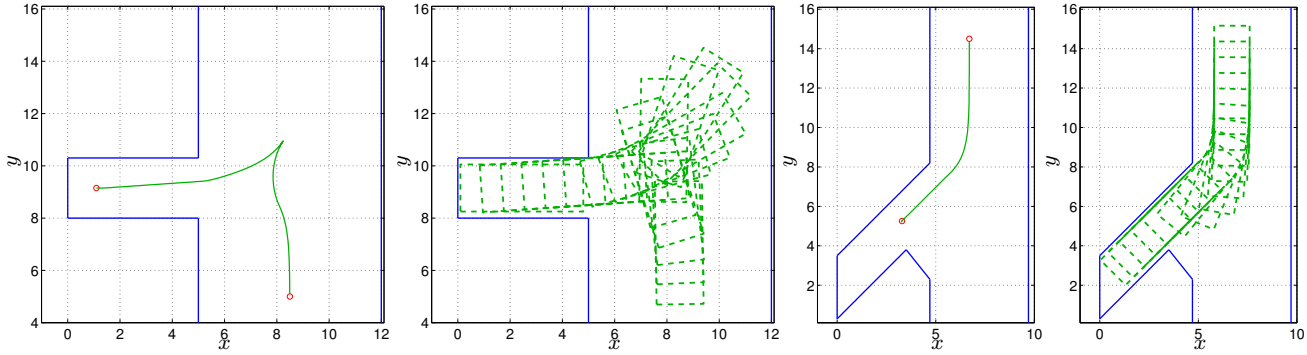

Fig. 7. Garage and angle parking scenario.

is very short.

V. Conclusion

In this paper, a fast optimization based motion planner for car parking is proposed. The path is discretized by means of a Runge-Kutta discretization and calculated by recurrently solving a local static optimization problem. The weighting terms for the optimization 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 phases and for direction changes is based on heuristic rules mimicking the behavior of a human driver.

Simulations for different scenarios show the feasibility of the proposed algorithm. Without any modifications parallel, garage and angle parking problems can be solved. 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 account for moving obstacles.

References

[1] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards." *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[2] M. F. Hsieh and U. Özgüner, "A parking algorithm for an autonomous vehicle," in *Proc. IEEE Intelligent Vehicles Symposium*, Eindhoven, Netherlands, Jun. 2008, pp. 1155–1160.

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

[4] D. Kim, W. Chung, and S. Park, "Practical motion planning for car-parking control in narrow environment," *IET Control Theory Applications*, vol. 4, no. 1, pp. 129–139, Jan. 2010.

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

[6] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[7] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," in *Proc. 5th Int. Conf. on Advanced Robotics*, vol. 2, Piscataway, NJ, Jun. 1991, pp. 1012–1017.

[8] I. Hussein and A. Bloch, "Optimal control of underactuated nonholonomic mechanical systems," *IEEE Trans. Autom. Control*, vol. 53, no. 3, pp. 668 –682, Apr. 2008.

[9] K. Kondak and 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.

[10] J.-P. Laumond, P. Jacobs, M. Taix, and R. Murray, "A motion planner for nonholonomic mobile robots," *IEEE J. Robot. Autom.*, vol. 10, no. 5, pp. 577 –593, Oct. 1994.

[11] S. Sekhavat and J.-P. Laumond, "Topological property for collision-free nonholonomic motion planning: the case of sinusoidal inputs for chained form systems," *IEEE J. Robot. Autom.*, vol. 14, no. 5, pp. 671–680, Oct. 1998.

[12] B. Müller, J. Deutscher, and S. Grodde, "Continuous curvature trajectory design and feedforward control for parking a car," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 541–553, May 2007.

[13] H. Liu, W. Liu, and L. Latecki, "Convex shape decomposition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Francisco, CA, Jun. 2010, pp. 97–104.

[14] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The Int. Journal of Robotic Research*, vol. 5, no. 3, pp. 72–89, Sep. 1986.

[15] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 108–120, Feb. 1983.

[16] F. Jean, "Complexity of nonholonomic motion planning," *Int. Journal of Control*, vol. 74, no. 8, pp. 776–782, May 2001.

[17] P. E. Gill, W. Murray, and M. A. Saunders, *Users Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, Dept. of Mathematics, University of California, San Diego, CA, Feb. 2006.