

Trajectory Generation and Feedforward Control for Parking a Car

Bernhard Müller, Joachim Deutscher and Stefan Grodde

Abstract—In this paper a two-step trajectory planning algorithm is applied to generate suitable trajectories for an autonomous parking maneuver of a car. It is shown how important requirements of the automotive industry can be met with the proposed approach. Furthermore, some details on the implementation of the algorithm are given, which are essential to obtain reasonable computation times.

I. INTRODUCTION

Currently, the automotive industry develops parking assistance systems supporting the driver during a parking maneuver. A trajectory planning algorithm, which generates a suitable collision-free path from a given start to a required goal position within the parking space that satisfies all kinematic constraints, is a central component of such a system. This kind of problem belongs to the most frequently considered examples in robotics literature, where various solution approaches are available, e.g. planning based on neural-networks, fuzzy-techniques, dynamic programming, numerical continuation methods and two-step algorithms making use of small-time controllability (see [1], [2], [3] and references therein). Nevertheless, rather empirical methods are applied for this task in practice (see e.g. [4], [5], [6]). One reason might be that many authors examined the case of completely autonomous maneuvering, whereas most versions of assistant systems only act on the steering mechanism of the car, such that the driver still has to control the velocity by himself. Another problem of the systematic techniques is that often complex computations are involved which are difficult to accomplish in real-time. Furthermore, some properties required by the automotive industry, like that the resulting trajectory should approximately reflect the behavior of a human driver or that steering at standstill should be avoided, are rather hard to integrate into the available systematic frameworks.

In this contribution a suitable version of a two-step algorithm is developed, which satisfies the requirements for a real-life application mentioned above. Moreover, the approach is based on a solid theory guaranteeing that a solution is always found provided that one exists. In principle, the proposed method is a combination of the collision-free planning algorithm developed in [7] and the trajectory planner for a car-like vehicle introduced in [8]. Having adapted the basic algorithms to the special requirements of the parking problem the condition that the velocity cannot be controlled

directly can be taken into account and the demanded qualitative properties of the resulting trajectories can be met. Furthermore, it is shown how reasonable computation times can be achieved by performing major parts of necessary expensive calculations offline. In this context, some details on a smart implementation structure of the algorithm are given.

In the next section an appropriate model of a car is introduced and the considered trajectory planning problem is formulated more precisely. Moreover, some basic consequences resulting from the special structure and fundamental properties of the car model are stated. In Section III the principles of the proposed two-step approach are explained, before the efficient implementation is addressed in Section IV. Finally, a simulation result is presented in Section V followed by some concluding remarks in Section VI.

II. PROBLEM STATEMENT AND PRELIMINARIES

A. Control Model

First, the problem stated during the introduction is specified more precisely applying basic tools and notions from robotics literature (see e.g. [1]). As shown in Figure 1 the

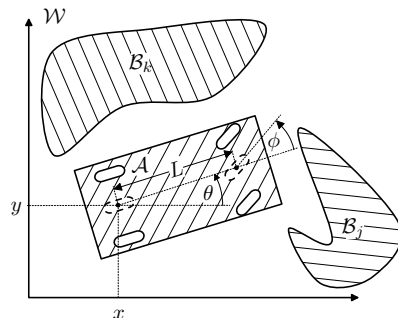


Fig. 1. Workspace \mathcal{W} with car body \mathcal{A} and obstacles \mathcal{B}_i

body of a car \mathcal{A} is approximated as a compact set of points within a rectangular region moving in a two-dimensional Euclidean space, which is referred to as *workspace* \mathcal{W} . Analogously, *obstacles* in the neighborhood of the parking space are represented as the compact sets of points \mathcal{B}_i , $i = 1, 2, \dots, m$. Assuming small velocities the dynamics of a car having axle base L can be described by a single-track model satisfying the state equations

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ \frac{1}{L} \tan \phi \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega \quad (1)$$

B. Müller and J. Deutscher are with the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, Cauerstrasse 7, 91058 Erlangen, Germany bernhard.mueller@rt.eei.uni-erlangen.de

S. Grodde wrote his diploma thesis at the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg while the paper was prepared.

with the state variables as defined in Figure 1. The inputs v and ω are the car velocity at the point $[x, y]^T$ and the angular steering velocity, respectively. Furthermore, the steering angle and angular velocity are constrained according to $|\phi| \leq \phi_{max} < \frac{\pi}{2}$ and $|\omega| \leq \omega_{max}$. The system description (1) can be further simplified by introducing the new state $\kappa = \frac{1}{L} \tan \phi$ and the new input $\sigma = \frac{1}{L \cos^2 \phi} \omega$ which yields

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ \kappa \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \sigma \quad (2)$$

with the *configuration*¹ vector $q = [x, y, \theta, \kappa]^T$ and the restrictions

$$|\kappa| \leq \kappa_{max} = \frac{1}{L} \tan \phi_{max} \quad (3)$$

$$|\sigma| \leq \sigma_{max} = \frac{1}{L} \omega_{max} \leq \frac{1}{L \cos^2 \phi} \omega_{max} \quad (4)$$

Note that in (4) the resulting state-dependent constraint of the new input σ is estimated by a lower state-independent bound, which is reasonable as long as $\cos^2 \phi \approx 1$, i.e. the upper bound of $|\phi|$ remains well beneath $\frac{\pi}{2}$. In order to simplify the statement of a collision condition the so-called *configuration obstacles*

$$\mathcal{CB}_i = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{B}_i \neq \emptyset\}, \quad i = 1, 2, \dots, m \quad (5)$$

are introduced according to [1], where $\mathcal{A}(q)$ means all points in \mathcal{W} covered by the body of the car at the configuration q . It should be mentioned that the configuration obstacles \mathcal{CB}_i are compact sets, since \mathcal{A} and \mathcal{B}_i are compact (see [1]).

B. Parking Problem

The problem considered in this paper can be formulated as follows.

Suppose that a collision-free start configuration q_s and a collision-free goal configuration q_g are given. Find feedforward control inputs $v_d(t)$ and $\sigma_d(t)$ steering the system (2) without violating the constraints (3)–(4) from $q(0) = q_s$ at time $t = 0$ to $q(T) = q_g$ at $t = T > 0$ such that

- (a) $q \cap \mathcal{CB}_i = \emptyset \forall t \in [0, T], \forall i = 1, 2, \dots, m$, and
- (b) $v_d(t)$ is piecewise constant with $|v_d| = v_{max}$.

Furthermore, the solution should satisfy the qualitative conditions that

- (c) the parking duration T is sufficiently small and
- (d) only few changes of the sign of $v_d(t)$ occur.

Note that the condition (b) of the list above is only stated for planning issues and does not represent unrealizable requirements for the implementation. It allows for the fact that the driver determines the absolute value of the velocity $v(t)$, meaning that the controller can only manipulate the second input σ and the points where the direction of motion

is changed, i.e. the sign of the first input v . In this case, if $v(t)$ can be measured and is assumed to be bounded by

$$|v(t)| \leq v_{max} \quad \forall t \in [0, T] \quad (6)$$

trajectory planning can be accomplished assuming the constant absolute value $|v_d| = v_{max}$. This can easily be seen by considering the problem in terms of arc-length s instead of time t , i.e. by using $s = s(t)$. With $|v| = \frac{ds}{dt}$ and by $v \neq 0$ the state equations (2) can be rewritten as

$$\begin{bmatrix} x'(t(s)) \\ y'(t(s)) \\ \theta'(t(s)) \\ \kappa'(t(s)) \end{bmatrix} = \begin{bmatrix} \cos \theta(t(s)) \\ \sin \theta(t(s)) \\ \kappa(t(s)) \\ 0 \end{bmatrix} \text{sign}(v(t(s))) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \bar{\sigma}(s) \quad (7)$$

where $\bar{\sigma}(s) = \frac{\sigma(t(s))}{|v(t(s))|}$ and the notational abbreviation $(\cdot)' = \frac{d(\cdot)}{ds}$ is used. Now assume that $\sigma_d(t)$ and $v_d(t)$ solving the parking problem are known, which implies $|v_d(t)| = v_{max}$ according to condition (b). This solution corresponds to the control

$$\bar{\sigma}_d(s) = \frac{\sigma_d(\frac{s}{v_{max}})}{v_{max}} \quad (8)$$

of the arc-length based model (7). In order to steer the vehicle along the same path with a different velocity $v(t)$, (8) must be retransformed to time resulting in the input

$$\tilde{\sigma}_d(t) = \bar{\sigma}_d(s(t)) |v(t)| = \sigma_d(\frac{s(t)}{v_{max}}) \frac{|v(t)|}{v_{max}} \quad (9)$$

with $s(t) = \int_0^t |v(\tau)| d\tau$, which can easily be implemented if $v(t)$ is measured. Since $|\tilde{\sigma}_d(t)| \leq |\sigma_d(t)|$ in view of (6) the constraint (4) is satisfied meaning that the resulting feedforward control (9) in combination with the driver-determined velocity $v(t)$ is admissible.

An important question is if and on what conditions a solution to the parking problem exists. As shown in [2] the system (2) including the constraints (3)–(4) and the restriction $|v(t)| = v_{max}$ is small-time controllable (see [2] for the definition of small-time controllability). As a consequence, if there is any collision-free connection in \mathcal{C} from q_s to q_g , there also exist feasible controls $\sigma_d(t)$ and $v_d(t)$ with $|v_d(t)| = v_{max}$ steering the system (2) from q_s to q_g along a collision-free trajectory. Hence, assuming realistic parking situations a solution can always be found as long as the parking space is at least as large as the body of the car. Obviously, however, the qualitative requirements on the necessary time T and the number of maneuvers (conditions (c) and (d)) cannot be met for very small parking spaces.

C. Flatness of the Car

Another very important property is the flatness (see [9]) of the unconstrained system (2). Roughly speaking flatness can be characterized by the existence of a flat output y_f with the same dimension as the input vector. If the trajectory for the flat output y_f is known, the trajectories x and u solving the system equations can be calculated by using algebraic equations. Thus, trajectory planning and feedforward control

¹As common in robotics the state space of (2) is called *configuration space* \mathcal{C} and its elements *configuration variables* throughout this paper.

problems can be solved algebraically for flat systems. In [10] it is shown that a flat output for (2) is given by

$$y_f = [y_{f1}, y_{f2}]^T = [x, y]^T \quad (10)$$

Consequently, all system variables can easily be calculated without integrating the differential equations if the evolution of the position $y_f(t) = [x(t), y(t)]^T$ and its derivatives with respect to time are known. This is proved within a general framework in [10], where a natural parametrization is used in order to avoid singularities and ambiguities. However, for the purpose of this paper it is convenient to assume that $v(t)$ is piecewise C^2 with $v^2(t) = \dot{x}^2(t) + \dot{y}^2(t) \neq 0 \forall t \in [0, T]$ and that the direction of motion $d_v(t) = \text{sign}(v(t))$ is known, which allows the straightforward derivation of the differential parametrization of the remaining states and inputs

$$\theta = d_v \text{atan}(\dot{y}_{f2}, \dot{y}_{f1}) \quad (11)$$

$$\kappa = \frac{\dot{y}_{f2}\dot{y}_{f1} - \ddot{y}_{f2}\dot{y}_{f1}}{(\dot{y}_{f1}^2 + \dot{y}_{f2}^2)^{\frac{3}{2}}} \quad (12)$$

$$v = d_v \sqrt{\dot{y}_{f1}^2 + \dot{y}_{f2}^2} \quad (13)$$

$$\sigma = \frac{(y_{f2}^{(3)}\dot{y}_{f1} - \dot{y}_{f2}y_{f1}^{(3)})(\dot{y}_{f1}^2 + \dot{y}_{f2}^2) - 3(\ddot{y}_{f2}\dot{y}_{f1} - \ddot{y}_{f1}\dot{y}_{f2})(\dot{y}_{f1}\ddot{y}_{f1} + \dot{y}_{f2}\ddot{y}_{f2})}{(\dot{y}_{f1}^2 + \dot{y}_{f2}^2)^{\frac{5}{2}}} \quad (14)$$

where $\text{atan}(\cdot, \cdot)$ is the four-quadrant inverse tangent function. Note that (11)–(14) are not defined on points of discontinuity of $v(t)$ (see condition (b) of the parking problem).

Thus, if a feasible reference trajectory $y_{fd}(t) = [x_d(t), y_d(t)]^T$ in the workspace \mathcal{W} is determined the corresponding evolution of the other configuration variables $\theta_d(t)$ and $\kappa_d(t)$ as well as the feedforward controls $v_d(t)$ and $\sigma_d(t)$ can readily be calculated by means of (11)–(14).

III. TWO STEP APPROACH

A. Motivation and Principle

Collision avoidance as well as steering of dynamical systems in the presence of input and state constraints are hard problems when taken separately. To avoid the difficulty of fulfilling both tasks simultaneously the solution presented in this paper is based on a two step approach (see e.g. [2], [11]). First, a collision-free curve connecting the start and goal configuration q_s and q_g is determined in the configuration space neglecting the system dynamics (2) and constraints (3)–(4). This so-called *path* can be written as a continuous map

$$\tau : [0, 1] \rightarrow \mathcal{C}, \quad \tau(0) = q_s, \quad \tau(1) = q_g \quad (15)$$

which satisfies the condition

$$\tau(\xi) \cap \mathcal{CB}_i = \emptyset, \quad \forall \xi \in [0, 1], \quad \forall i = 1, 2, \dots, m \quad (16)$$

for collision avoidance. Second, a feasible trajectory $y_{f,d}(t) = [y_{f1,d}(t), y_{f2,d}(t)]^T$ for the flat output is planned by moving roughly along the collision-free path at the piecewise constant velocity $|v| = v_{max}$ (see condition (b)). Recall that any trajectory planned for the flat output in \mathcal{W} can easily be converted into a corresponding trajectory $q_d(t)$

in \mathcal{C} by means of (11)–(12). Within this second phase the obstacles are not considered explicitly. However, in principle it is always possible to find an obstacle-free trajectory using this approach, since the small-time controllability property (see Section II-B) guarantees that τ can be approximated by a feasible trajectory staying in an arbitrary close neighborhood of τ in \mathcal{C} . Moreover, as the configuration obstacles are assumed to be compact, every point of τ is surrounded by an obstacle-free neighborhood.

The conversion of the collision-free path τ into a feasible trajectory can be implemented using the following iterative algorithm (see [2]). At initialization stage a suitable number of p ascending sampling points

$$\bar{q}_i = \tau(\xi_i), \quad 0 = \xi_1 < \xi_2 < \dots < \xi_p = 1, \quad i = 1, \dots, p \quad (17)$$

on the obstacle-free path τ connecting q_s and q_g is chosen. Ignoring the obstacles each pair of successive configurations \bar{q}_i and \bar{q}_{i+1} is connected by a feasible trajectory $y_{f,d}(t)$ satisfying the state and input constraints (3)–(4). For that task a so-called *local trajectory planner* *Steer* is needed which must be able to calculate a connecting trajectory between any two configurations (see Section III-C), i.e. *Steer* can be interpreted as a map

$$\text{Steer} : \mathcal{C} \times \mathcal{C} \times [t_1, t_2] \rightarrow \mathcal{C} \quad (18)$$

such that $\text{Steer}(q_1, q_2, t_1) = q_1$, $\text{Steer}(q_1, q_2, t_2) = q_2$ and the continuous curve $\text{Steer}(q_1, q_2, t)$, $t \in [t_1, t_2]$, represents a feasible trajectory with respect to (2) and the constraints (3)–(4). Next, the resulting sub-trajectories are tested for collisions with obstacles. If a sub-trajectory connecting \bar{q}_i and \bar{q}_{i+1} is not collision-free an additional sampling point between \bar{q}_i and \bar{q}_{i+1} on τ is added such that the distance of successive configurations is reduced. The underlying idea is that the connecting trajectory of two near configurations stays closer to τ than the connection of two configurations located far away of each other. This expectation is well-founded if the algorithm *Steer* satisfies the *topological property*

$$\forall \epsilon > 0, \quad \exists \eta > 0, \quad \forall q_1, q_2 \in \mathcal{C}$$

$$\|q_2 - q_1\| < \eta \Rightarrow \|\text{Steer}(q_1, q_2, t) - q_1\| < \epsilon \quad \forall t \in [t_1, t_2] \quad (19)$$

(see [2]), where $\|\cdot\|$ means any norm defined on \mathcal{C} . In fact, it is shown in [2], [12] that a collision-free path can be tracked arbitrarily close using the outlined procedure in combination with a local trajectory planner fulfilling (19). Thus, the proposed algorithm which converts an obstacle-free path τ into an obstacle-free feasible trajectory is complete provided that the employed local trajectory planner satisfies the topological property. As a consequence, the described two step approach always results in a solution to the parking problem assuming that one exists and that the algorithm used for determining an obstacle-free path is complete.

B. Step 1: Obstacle Distance Optimized Path Planner for Generating a Collision-free Path

So far nothing was said about how the qualitative conditions (c) and (d) of the parking problem can be taken

into account. In fact, the required properties can be achieved by choosing suitable solution methods for the two separate steps of the algorithm. First, condition (d) is considered, i.e. the avoidance of unnecessary shifts in direction. As shown in [12], [7] the number of maneuvers generally is reduced, if only few sampling points on the collision-free path are needed. Following the ideas of [7], a reduction of the number of necessary sampling points can be obtained by planning the collision-free path τ in \mathcal{C} such that it stays as far away from the obstacles as possible since this assures maximum free space for the succeeding trajectory evaluation.

For this task a measure of distance on the configuration space \mathcal{C} is required and it is easily checked that the convenient Euclidean metric is not appropriate. Instead, the *shortest feasible path (SFP) metric* (see [7], [13])

$$d_{SFP} : \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}_0^+ \quad (20)$$

is introduced being defined as the arc-length in \mathcal{W} of the shortest feasible connecting trajectory between two configurations. Figure 2 shows the motivation for this definition, which implicitly takes the restrictions of motion of the vehicle into account: The SFP distance $d_{SFP}(q, q_1)$ to a destination q_1 requiring a sideway motion is large whereas a small distance value $d_{SFP}(q, q_2)$ is assigned to a destination q_2 which can easily be reached from q without maneuvering. The introduced concept can easily be extended to describe

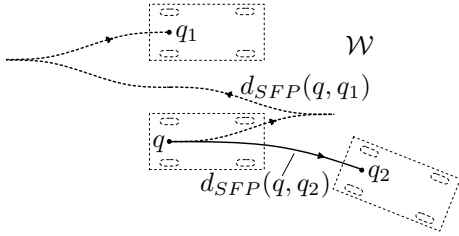


Fig. 2. Definition of the SFP metric

the SFP distance $d_{SFP}(q, \mathcal{CB}_i)$ between a single configuration q and a configuration obstacle \mathcal{CB}_i which can be expressed as

$$d_{SFP}(q, \mathcal{CB}_i) = \min_{\hat{q} \in \partial \mathcal{CB}_i} d_{SFP}(q, \hat{q}) \quad (21)$$

where $\partial \mathcal{CB}_i$ is the set of boundary points of \mathcal{CB}_i and it is assumed that all configuration obstacles \mathcal{CB}_i are compact (see [1]). In addition, the SFP distance of q to the entire obstacle distribution \mathcal{CB} is introduced as

$$d_{SFP}(q, \mathcal{CB}) = \min_i d_{SFP}(q, \mathcal{CB}_i) \quad (22)$$

for notational convenience.

Using the SFP metric the obstacle distance optimized path planning algorithm, which is illustrated for a parallel parking scheme in Figure 3, can be roughly described as follows. Starting from the start configuration q_s and the goal configuration q_g continuous subpaths τ_s and τ_g with $\tau_s(0) = q_s$ and $\tau_g(0) = q_g$ are planned, respectively, such that the minimum distance to all obstacles is (locally) maximized and that one moves roughly into a given priority direction. After τ_s and τ_g have reached a prespecified region

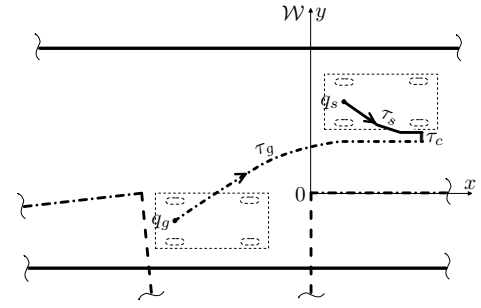


Fig. 3. Algorithm for distance optimized path planning

of \mathcal{C} the subpaths are combined to an overall path τ from q_s to q_g either by using an intersection point between τ_s and τ_g or by adding a particular connecting subpath τ_c calculated on the basis of the symmetry of the obstacle distribution. The details of the algorithm are given in [12].

Obviously, the algorithm outlined above can only be implemented if the effort to calculate the SFP distance to obstacles according to (21) is limited. Even the actual computation of the SFP distance between two single configurations, however, implies the evaluation of the shortest feasible trajectory with respect to the vehicle model (2) and the constraints (3)–(4). Unfortunately, this task still represents an unsolved problem in the optimal control literature (see e.g. [14]). A possible remedy is to use the SFP distance induced by removing the restriction (4), i.e. allowing arbitrarily fast steering by setting $\sigma_{max} \rightarrow \infty$. On that condition Reeds and Shepp [15] showed that the corresponding shortest feasible trajectory for the flat output y_f consists of a concatenation of straight-line segments and arcs of circles in \mathcal{W} . Furthermore, they proved that the shortest trajectories can be classified in 48 three-parameter families. For the sake of completeness it should be mentioned that Sussmann and Tang [16] could reduce the number of sufficient families to 46. In principle, the calculation of the SFP distance for the case $\sigma_{max} \rightarrow \infty$ can now be accomplished as follows. Given two configurations one can determine the three defining parameters for each of the 48 trajectory candidates, calculate their arc-lengths and take the minimum arc-length as the required SFP distance. The actual computational complexity is somewhat lower since the evaluation of only 9 trajectories is sufficient due to symmetry properties. The details of the calculation of the SFP metric can be found in [13]. Note that being able to compute the distance between two single configurations the evaluation of the obstacle distance according to (21) is also feasible but computational expensive. However, as will become clear in Section IV-A most of the expensive calculations can be accomplished offline.

C. Step 2: Continuous Curvature Trajectory Planner for Generating a Feasible Trajectory

A small parking duration T as required in the qualitative condition (c) of the parking problem can be achieved by using a suitable algorithm for the trajectory generation. The core of this second step is formed by the *continuous-curvature (CC) local trajectory planner* $Steer_{cc}$ proposed in

[8]. By construction this particular algorithm yields short trajectories, although no strict optimality condition is satisfied. Moreover, it fulfills the state and the input restrictions (3)–(4) explicitly and can easily be extended in order to satisfy the topological property (see Section III-B). For details the reader is referred to [8], [17].

IV. IMPLEMENTATION

Focusing on the first step of the outlined algorithm (see Section III-B) it is shown in the sequel, how the most computational expensive calculations can be accomplished offline. Furthermore, some details on the numerical implementation of the method are given.

A. Obstacle Distribution

As explained in Section III-B the actual evaluation of the SFP obstacle distance d_{SFP} is very computational expensive. Thus, it is desirable to accomplish as much of the necessary calculations offline as possible, which is only feasible, if the number of considered obstacle distributions can be restricted without losing significant amounts of generality. In case of the parking problem, a straight line $\partial\mathcal{B}_L$ and a half straight line $\partial\mathcal{B}_{HL}$ are chosen as basic elements of obstacle boundaries in \mathcal{W} . Recall that, since all obstacles are assumed to be compact, the definition of their boundaries is sufficient (see (21)). Figure 4 illustrates how typical parking situations can be approximated using these simple geometrical components. Assuming simplified geometrical

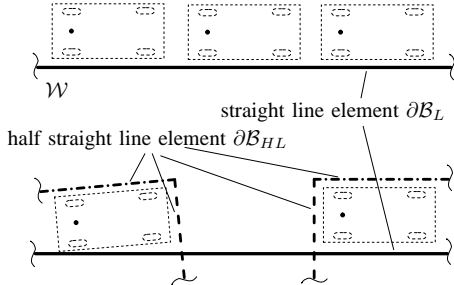


Fig. 4. Approximation of obstacle distributions using straight line ($\partial\mathcal{B}_L$) and half straight line ($\partial\mathcal{B}_{HL}$) boundary sets

obstacle distributions of this form the computation of the SFP obstacle distance can be accomplished in the following steps. First, the SFP distance values to a reference half straight line and a reference straight line obstacle are stored in look-up tables, respectively, for all car configurations on a sufficient narrow grid of sampling points within a significant range. This computational expensive step can be performed offline. Second, the measured shape of a specific parking space is estimated by combining line segments representing shifted and rotated copies of the reference elements (see Figure 4). In terms of the look-up tables shifting and rotating can easily be taken into account by shifting indices. Thus, the SFP distance of a given car configuration to the entire obstacle distribution can be determined by calculating the corresponding indices and taking the minimum value of the looked-up distances to all basic elements. Only this second step, which mainly

consists of looking-up a small number of values in memory, has to be executed online.

B. Symmetry of the Basic Obstacle Elements

It is obvious that the reduction of online computing time faces considerably higher memory requirements since without further measures two 3-dimensional look-up tables must be stored. Note that by setting $\sigma_{max} \rightarrow \infty$ the SFP obstacle distance becomes independent of σ , i.e. the fourth dimension can be omitted in the first place. However, the amount of necessary memory can be reduced significantly by using the symmetry properties of the basic boundary elements. First, focus on the reference straight line element $\partial\mathcal{B}_L$ illustrated on the left of Figure 5. It is immediately ap-

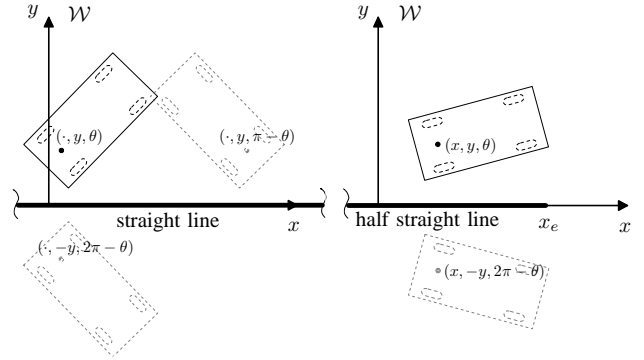


Fig. 5. Symmetry properties of the basic obstacle elements

parent that the corresponding SFP distance $d_{SFP}^L(x, y, \theta) = d_{SFP}^L((x, y, \theta)^T, \partial\mathcal{B}_L)$ does not depend on x meaning that a two-dimensional look-up table is sufficient. Furthermore, the relations

$$d_{SFP}^L(x, y, \theta) = d_{SFP}^L(x, y, \pi - \theta) \quad (23)$$

and

$$d_{SFP}^L(x, y, \theta) = d_{SFP}^L(x, -y, 2\pi - \theta) \quad (24)$$

hold which allow the restriction to positive angle values $0 \leq \theta < \pi$ and positive y -coordinates $y \geq 0$. The symmetry of the reference half straight line segment $\partial\mathcal{B}_{HL}$ is indicated on the right of Figure 5. Obviously, the SFP distance $d_{SFP}^{HL}(x, y, \theta) = d_{SFP}^{HL}((x, y, \theta)^T, \partial\mathcal{B}_{HL})$ satisfies

$$d_{SFP}^{HL}(x, y, \theta) = d_{SFP}^{HL}(x, -y, 2\pi - \theta) \quad (25)$$

which means that the values for $y < 0$ can be omitted. In addition, in regions where $x \ll x_e$ the endpoint has no impact, i.e. the relationship

$$d_{SFP}^{HL}(x, y, \theta) = d_{SFP}^L(x, y, \theta) \text{ for } x \ll x_e \quad (26)$$

permits the truncation of small x values in the look-up table corresponding to $\partial\mathcal{B}_{HL}$.

C. Numerical Calculation of Distance Look-up Tables

Assuming that an algorithm for the SFP distance evaluation $d_{SFP}(q_1, q_2)$ between two discrete configurations q_1 and q_2 is available (see Section III-B), the numerical calculation of the SFP obstacle distance $d_{SFP}(q, \partial\mathcal{C}\mathcal{B}_i)$ between a configuration q and an obstacle $\mathcal{C}\mathcal{B}_i$ can be

accomplished by means of a direct numerical implementation of expression (21). The computation is divided into two parts: First, discrete points on the boundary of the configuration obstacle $\partial\mathcal{CB}_i$ are determined. Second, the SFP distances to all discrete configurations on the obstacle boundary are calculated and the minimum operation is applied. The actual realization of the first step is explained using the example of a half straight line obstacle element $\partial\mathcal{B}_{HL}$. Recall that the boundary of a configuration obstacle is defined by all configurations q where the body of the car touches the obstacle. Thus, the basic idea is to determine the obstacle boundary by recording all configurations while moving and turning the car in the workspace \mathcal{W} such that its body remains in touch with the obstacle. Due to the fact that the rectangular-shaped car body represents a convex polygon and the obstacle distribution is a polygonal region (see [1]), only eight possible contacts between the car body and the obstacle can occur, namely that one of the four vertices of the car lies on the half straight line element or that the vertex of the half straight line lies on one of the four edges of the car. Since each case can be described by simple geometrical conditions (being omitted for lack of space), discrete configurations of $\partial\mathcal{CB}_{HL}$ can be scanned with respect to a suitable discretization grid by the outlined fundamental procedure. The second step for actually evaluating $d_{SFP}(q, \partial\mathcal{CB}_{HL})$ is straightforward.

V. SIMULATION RESULTS

The proposed approach was implemented in MATLAB and tested for several different sets of car parameters and various parking situations. For the parallel parking situation shown in Figure 6 car parameters corresponding approximately to the configuration of a VW Golf are assumed. Furthermore, the maximum steering velocity $\omega_{max} =$

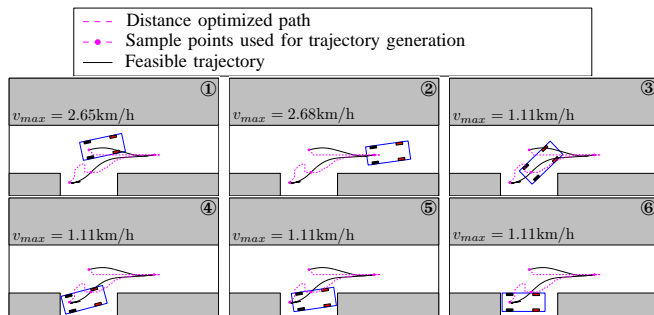


Fig. 6. Distance optimized path and resulting trajectory for a parking space, which is 1.4m longer and 0.4m wider than the body of the car.

0.43rad/s is chosen such that the steering angle can be changed from maximum left to maximum right in about 3 seconds. The simulation result verifies that the algorithm is able to handle tiny parking spaces as well as unsuitable start configurations. Moreover, the resulting trajectories are natural, i.e. they are similar to the behavior of a human driver.

Using the current implementation in MATLAB, the total computation time for the example in Figure 6 was about 2.5s on an AMD Athlon XP 1800+. However, there still

is a great potential for optimizing the computing time left unfulfilled, especially with respect to the realization of the second step of the algorithm (see Section III-C). In view of this aspect and the fact that MATLAB is a resources consuming platform, suitable calculation times of less than 1s seem feasible with the proposed approach. At the same time, the memory requirements are rather moderate. For instance, the obstacle distance look-up tables applied for the example in Figure 6 require a total of about 2.5MByte RAM using the sufficient numerical precision of one byte per value.

VI. CONCLUSIONS

In this contribution a two-step algorithm generating suitable trajectories for an autonomous parking maneuver was presented. Apart from the basics of the proposed procedure, some aspects of its efficient implementation were addressed in detail. In particular, it was shown that using appropriate approximations of realistic parking environments expensive calculations can be accomplished offline.

REFERENCES

- [1] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [2] J. Laumond, S. Sekhavat, and F. Lamiroux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*, J. Laumond, Ed. Springer, 1998.
- [3] I. Kolmanovsky and N. McClamroch, "Developments in nonholonomic control problems," *IEEE Control Systems Magazine*, vol. 15, no. 6, pp. 20–36, 1995.
- [4] J. Shyu and C. Chuang, "Automatic parking device for automobile," U.S. patent no. 4931930, 1990.
- [5] I. Paromtchik and C. Laugier, "Motion generation and control for parking an autonomous vehicle," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 3117–3122.
- [6] M. Wada, K. Yoon, and H. Hashimoto, "Development of advanced parking assistance system," *IEEE Trans. on Industrial Electronics*, vol. 50, no. 1, 2003.
- [7] B. Mirtich and J. Canny, "Using skeletons for nonholonomic path planning among obstacles," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1992, pp. 2533–2540.
- [8] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 6, 2004.
- [9] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "A Lie-Bäcklund approach to equivalence and fitness of nonlinear systems," *Trans. Aut. Control*, vol. 61, pp. 1337–1361, 1999.
- [10] P. Rouchon, M. Fliess, J. Lévine, and P. Martin, "Flatness and motion planning: the car with n trailers," in *Proc. of the European Control Conf.*, 1993, pp. 1518–1522.
- [11] J. Laumond, "Feasible trajectories for mobile robots with kinematic and environment constraints," in *Proc. of the Int. Conf. on Intelligent Autonomous Systems*, 1986, pp. 346–354.
- [12] B. Müller, J. Deutscher, and S. Grodde, "Continuous curvature trajectory design and feedforward control for parking a car," *submitted to IEEE Trans. on Control Systems Technology*, 2006.
- [13] J. Laumond and P. Souères, "Metric induced by the shortest paths for a car-like mobile robot," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 1993, pp. 1299–1303.
- [14] H. Sussmann, "The Markov-Dubins problem with angular acceleration control," in *Proc. of the 36th IEEE Conf. on Decision and Control*, 1997, pp. 2639–2643.
- [15] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forward and backward," *Pacific Journal of Mathematics*, vol. 145, no. 2, 1990.
- [16] H. Sussmann and W. Tang, "Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control," Rutgers University, Report SYCON-91-10, 1991.
- [17] S. Grodde, "Lokaler Trajektorienplaner für das automatische Einparken," Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, technical report, 2005.