Continuous Curvature Trajectory Design and Feedforward Control for Parking a Car

Bernhard Müller, Joachim Deutscher, and Stefan Grodde

***Abstract*—In this paper, a two-step trajectory planning algorithm from robotics literature is applied to generate suitable trajectories for an autonomous parking maneuver of a car. First, a collision-free curve between a given start and a desired goal configuration within the parking space is planned ignoring the kinematic restrictions on the movement of the car. Second, the collision-free curve is converted into a feasible collision-free trajectory, which can be exactly followed by the car. It is shown how this general planning scheme must be adapted to meet the requirements of the automotive industry.**

***Index Terms*—Autonomous parking, feedforward control, flat systems, motion-planning, nonholonomic systems.**

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 task belongs to the most frequently considered benchmark problems for nonholonomic motion planning algorithms in the presence of obstacles, where various solutions are available in robotics literature. For example, Laumond *et al.* [1] introduced a two-step algorithm for small-time controllable systems. Barraquand and Latombe [2] applied a best first search algorithm to a discretized version of the problem. Divelbiss and Wen [3] used a numerical continuation method, which was also analyzed from a theoretical view by Sussmann and Chitour [4], [5]. Many authors considered intelligent control approaches like fuzzy-techniques (see [6] and [7]) or neural networks (see [8]). A thorough literature review can be found in [9]–[11].

In spite of the variety of available approaches, rather empirical methods are applied for the considered task in practice (see e.g., [12]–[14]). 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 realtime. 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 according to [1] is developed, which satisfies the requirements for a real-life application mentioned previously. 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 [15] and the trajectory planner for a car-like vehicle introduced in [16]. The basic procedure is as follows. In the first step, a collision-free curve connecting the start and the goal configuration is planned without considering the kinematic restrictions on the movement of the car. The second step computes a feasible collision-free trajectory by moving roughly along that collision-free curve. Having adapted this basic algorithm 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. To this end, an adaptive upper bound for the maximum allowed velocity during the parking maneuver is proposed, which enables the application of the approach to very small parking spaces. Moreover, it is shown how the computation time can be reduced by performing major parts of necessary expensive calculations offline.

In Section II, 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 the flatness property of the car model are stated. In Section III, the principles of the proposed two-step approach are explained and the completeness of the algorithm is analyzed. The main results of this contribution follow in Section IV, where the adaption of the fundamental two-step planning scheme to the special requirements of the parking problem are addressed and the algorithms used within the two single steps are explained in detail. In Section V, some significant implementation aspects are stated, before the performance of the algorithm is verified by means of simulation results 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., [9]). As shown in Fig. 1, the *body of a car* $\mathcal{A}$ is approximated as a compact set of points within a rectangular region moving in a 2-D Euclidean space, which is referred to as

Manuscript received July 20, 2006. Manuscript received in final form December 7, 2006. Recommended by Associate Editor I. Kolmanovsky.

The authors are with the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, 91058 Erlangen, Germany (e-mail: bernhard.mueller@rt.eei.uni-erlangen.de; joachim.deutscher@rt.eei.uni-erlangen.de; stefan_grodde@web.de).

Color versions of Figs. 5, 11, and 12 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2006.890289

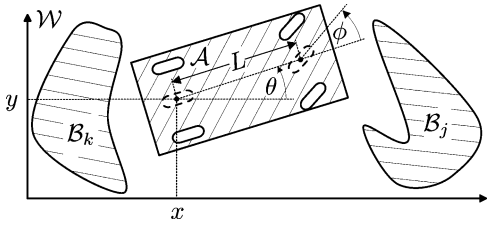


Fig. 1. Workspace $\mathcal{W}$ with car body $\mathcal{A}$ and obstacles $\mathcal{B}_i$.

workspace $\mathcal{W}$. Analogously, *obstacles* in the neighborhood of the parking space are represented as the compact sets of points $\mathcal{B}_i, i = 1, 2, \ldots, m$. Assuming small velocities, the dynamics of a car having axle base L can be described by a single-track model satisfying

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

where $[x, y]^T$ defines the position of the midpoint of the rear axle in a Euclidean reference frame, θ means the orientation of the car relative to the x-axis and ϕ is the steering angle. 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 restricted according to

$$|\phi| \leq \phi_{\max} < \frac{\pi}{2}, \quad |\omega| \leq \omega_{\max} \tag{2}$$

due to mechanical constraints and the properties of the steering actuator. The system description (1) can be further simplified by introducing $\kappa = (1/L)\tan\phi$ and $\sigma = \omega/L\cos^2\phi$ which yields

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \sigma \tag{3}$$

with the *configuration*[1] *vector* $q = [x, y, \theta, \kappa]^T$ and the restrictions [see (2)]

$$|\kappa| \leq \kappa_{\max} = \frac{1}{L}\tan\phi_{\max} \tag{4}$$

$$|\sigma| \leq \sigma_{\max} = \frac{1}{L}\omega_{\max} \leq \frac{1}{L\cos^2\phi}\omega_{\max}. \tag{5}$$

Note that in (5) 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., if $|\phi|$ [see (2)] remains well beneath $\pi/2$. In order to simplify the statement of a collision condition, the so-called *configuration obstacles*

$$\mathcal{CB}_i = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{B}_i \neq \emptyset\}, \quad i = 1, 2, \ldots, m \tag{6}$$

are introduced according to [9], 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 [9]).

[1]As common in robotics, the state space of (3) is called *configuration space* $\mathcal{C}$ and its elements *configuration variables* throughout this paper.

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 (3) without violating the constraints (4) and (5) from $q(0) = q_s$ at time $t = 0$ to $q(T) = q_g$ at $t = T > 0$ such that the following:

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

Furthermore, the solution should satisfy the qualitative conditions that follow:

3) the parking duration T is sufficiently small;
4) only few changes of the sign of $v_d(t)$ occur.

Note that the condition 2) of the previous list 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

$$|v(t)| \leq v_{\max} \forall t \in [0, T] \tag{7}$$

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

$$|v| = \frac{ds}{dt} \tag{8}$$

and by $v \neq 0$ the state equation (3) can be rewritten as

$$\begin{bmatrix} x'(t(s)) \\ y'(t(s)) \\ \theta'(t(s)) \\ \kappa'(t(s)) \end{bmatrix} = \begin{bmatrix} \cos\theta(t(s)) \\ \sin\theta(t(s)) \\ \kappa(t(s)) \\ 0 \end{bmatrix} \operatorname{sign}(v(t(s))) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \bar{\sigma}(s) \tag{9}$$

where

$$\bar{\sigma}(s) = \frac{\sigma(t(s))}{|v(t(s))|} \tag{10}$$

and the notational abbreviation $(\cdot)' = d(\cdot)/ds$ 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 2). In view of (8) and (10), this solution corresponds to the control

$$\bar{\sigma}_d(s) = \frac{\sigma_d\left(\frac{s}{v_{\max}}\right)}{v_{\max}} \tag{11}$$

of the arc-length-based model (9). In order to steer the vehicle along the same path with a different velocity $v(t)$, (11) must be retransformed to time resulting in the input

$$\tilde{\sigma}_d(t) = \bar{\sigma}_d(s(t))|v(t)| = \sigma_d\left(\frac{s(t)}{v_{\max}}\right)\frac{|v(t)|}{v_{\max}} \tag{12}$$

[see (8) and (10)] 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 (7), the constraint (5) is satisfied meaning that the resulting feedforward control (12) in combination with the driver-determined velocity $v(t)$ is admissible.

C. Solvability of the Parking Problem

After the problem is stated, an important question is if and on what conditions a solution exists. The answer, as given in [10], is briefly recalled here. A system is called *small-time controllable* if the set of points reachable from any configuration $q \in \mathcal{C}$ before a given time T contains a neighborhood of q for all T. A descriptive interpretation of this definition is that assuming a compact space of admissible inputs a small-time controllable system can be steered approximately along any connected curve in the configuration space $\mathcal{C}$ without leaving a given, possibly very small, neighborhood around this curve. As shown in [10], the system (3), including the constraints (4) and (5) and the restriction $|v(t)| = v_{\max}$, is small-time controllable. Consequently, 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 (3) 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 larger than the body of the car.

D. Flatness of the Car

Another very important property is the flatness (see [17]) of the unconstrained system (3). 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 problems can be solved algebraically for flat systems. In [18], it is shown that a flat output for (3) is given by

$$y_f = [y_{f1}, y_{f2}]^T = [x, y]^T. \tag{13}$$

As a consequence, 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 proven within a general framework in [18], 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 $d_v(t) = \operatorname{sign}(v(t))$ is known, which allows the straightforward derivation of the differential parametrization of the remaining states and inputs shown in the following and (17) at the bottom of the page:

$$\theta = \operatorname{atan}(\dot{y}_{f2}, \dot{y}_{f1}) + \frac{1}{2}(d_v - 1)\pi \tag{14}$$

$$\kappa = \frac{\ddot{y}_{f2}\dot{y}_{f1} - \dot{y}_{f2}\ddot{y}_{f1}}{\left(\dot{y}_{f1}^2 + \dot{y}_{f2}^2\right)^{\frac{3}{2}}} \tag{15}$$

$$v = d_v\sqrt{\dot{y}_{f1}^2 + \dot{y}_{f2}^2} \tag{16}$$

where $\operatorname{atan}(\cdot, \cdot)$ is the four-quadrant inverse tangent function. Note that (14)–(17) are not defined on points of discontinuity of $v(t)$. However, as the condition 2) of the parking problem is only stated for planning issues and $v(t)$ is always continuous in reality, this does not imply any restrictions.

Thus, if in the workspace $\mathcal{W}$ a feasible reference trajectory $y_{fd}(t) = [x_d(t), y_d(t)]^T$ 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 (14)–(17).

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., [1], [10], and [19]). 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 (3) and constraints (4) and (5). This so-called *path* can be written as a continuous map

$$\tau : [0, 1] \to \mathcal{C}, \tau(0) = q_s, \tau(1) = q_g \tag{18}$$

which satisfies the condition

$$\tau(\xi) \cap \mathcal{CB}_i = \emptyset \, \forall \xi \in [0, 1] \, \forall i = 1, 2, \ldots, m \tag{19}$$

for collision avoidance. Second, a feasible trajectory

$$y_{f,d}(t) = [y_{f1,d}(t), y_{f2,d}(t)]^T = [x_d(t), y_d(t)]^T \tag{20}$$

for the flat output is planned by moving roughly along the collision-free path at the piecewise constant velocity $|v| = v_{\max}$ [see condition 2)]. 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 (14) and (15). Within this second phase, the obstacles are not considered explicitly. However, in principle, it is always possible to find an obstacle-free trajectory

$$\sigma = \frac{\left(y_{f2}^{(3)}\dot{y}_{f1} - \dot{y}_{f2}y_{f1}^{(3)}\right)\left(\dot{y}_{f1}^2 + \dot{y}_{f2}^2\right) - 3(\ddot{y}_{f2}\dot{y}_{f1} - \ddot{y}_{f1}\dot{y}_{f2})(\dot{y}_{f1}\ddot{y}_{f1} + \dot{y}_{f2}\ddot{y}_{f2})}{\left(\dot{y}_{f1}^2 + \dot{y}_{f2}^2\right)^{\frac{5}{2}}} \tag{17}$$

using this approach, since the small-time controllability property (see Section II-C) 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.

B. Basic Algorithm

The outlined procedure can be implemented using the following iterative algorithm (see [10]), which will be explained in detail for the considered application in Section IV-C. At initialization stage, a suitable number of p ascending sampling points

$$\bar{q}_i = \tau(\xi_i), 0 = \xi_1 < \xi_2 < \cdots < \xi_p = 1, i = 1, \ldots, p \quad (21)$$

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 $y_{f,d}(t)$ [see (20)] satisfying the state and input constraints (4) and (5). 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-E), i.e., Steer can be interpreted as a map

$$\text{Steer} : \mathcal{C} \times \mathcal{C} \times [t_1, t_2] \to \mathcal{C} \quad (22)$$

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 (3) and the constraints (4) and (5). Next, the resulting subtrajectories are tested for collisions with obstacles. If a subtrajectory 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.

C. Completeness of the Algorithm

The underlying idea for adding more sampling points in the case of a detected collision 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. In fact, this expectation is well-founded if the algorithm Steer used for trajectory generation satisfies the *topological property*

$$\forall \epsilon > 0, \exists \eta > 0, \forall q_1, q_2 \in \mathcal{C}$$
$$||q_2 - q_1|| < \eta \Rightarrow ||\text{Steer}(q_1, q_2, t) - q_1|| < \epsilon \, \forall t \in [t_1, t_2] \quad (23)$$

(see [10]), where $\|\cdot\|$ means any norm defined on $\mathcal{C}$. Fig. 2 shows the meaning of the topological property. Suppose that an ϵ-neighborhood of every point on τ is obstacle-free in $\mathcal{C}$. Then the connecting trajectory $\text{Steer}(q_1, q_2, t)$, $t \in [t_1, t_2]$, of all configurations q_1 and q_2 on τ which are not separated further than an η-distance of each other, i.e., $||q_2 - q_1|| < \eta$, stays within this collision-free ϵ-neighborhood. 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.

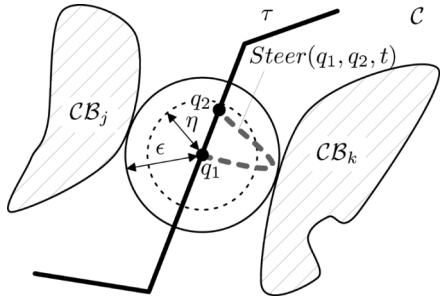


Fig. 2. Topological property.

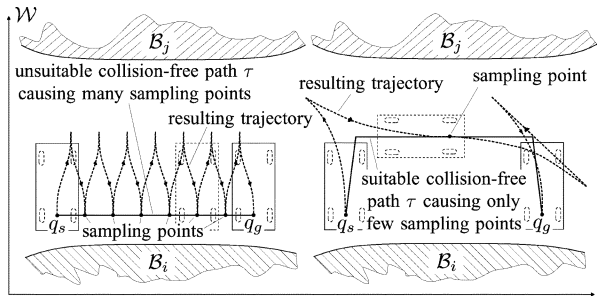


Fig. 3. Motivation for obstacle distance optimized path planning: Reducing the number of sampling points on τ results in fewer and less complex maneuvers.

D. Step 1: Obstacle Distance Optimized Path Planner for Generating a Collision-Free Path

So far nothing was said about how the qualitative conditions 3) and 4) 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 4) is considered, i.e., the avoidance of unnecessary shifts in direction. As illustrated in Fig. 3, the number of maneuvers generally is reduced, if only few sampling points on the collision-free path are needed. Following the ideas of [15], 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 [15] and [20])

$$d_{\text{SFP}} : \mathcal{C} \times \mathcal{C} \to \mathbb{R}_0^+ \quad (24)$$

is introduced being defined as the arc-length in $\mathcal{W}$ of the shortest feasible connecting trajectory between two configurations. Fig. 4 shows the motivation for this definition, which implicitly takes the restrictions of motion of the vehicle into account: The SFP distance $d_{\text{SFP}}(q, q_1)$ to a destination q_1 requiring a sideway motion is large, whereas a small distance value $d_{\text{SFP}}(q, q_2)$ is assigned to a destination q_2 which can easily be reached from q without maneuvering. The introduced

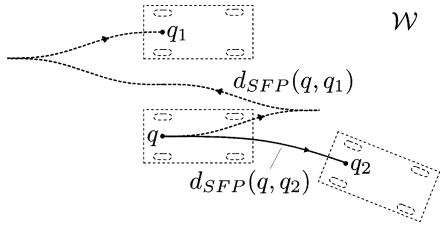


Fig. 4. Definition of the SFP metric.

concept can easily be extended to describe the SFP distance $d_{\mathrm{SFP}}(q, \mathcal{CB}_i)$ between a single configuration q and a configuration obstacle $\mathcal{CB}_i$ which is defined as

$$d_{\mathrm{SFP}}(q, \mathcal{CB}_i) = \inf_{\hat{q} \in \mathcal{CB}_i} d_{\mathrm{SFP}}(q, \hat{q}). \tag{25}$$

Since all considered configuration obstacles $\mathcal{CB}_i$ are assumed to be compact, (25) can be further simplified to (see [9])

$$d_{\mathrm{SFP}}(q, \mathcal{CB}_i) = \min_{\hat{q} \in \partial \mathcal{CB}_i} d_{\mathrm{SFP}}(q, \hat{q}) \tag{26}$$

where $\partial \mathcal{CB}_i$ is the set of boundary points of $\mathcal{CB}_i$. In addition, the SFP distance of q to the entire obstacle distribution $\mathcal{CB}$ is introduced as

$$d_{\mathrm{SFP}}(q, \mathcal{CB}) = \min_i d_{\mathrm{SFP}}(q, \mathcal{CB}_i). \tag{27}$$

Using the SFP metric, the obstacle distance optimized path planning algorithm 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 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 applying a particular connecting procedure based on the symmetry of the obstacle distribution. Further details are given in Section IV-B.

Obviously, the procedure outlined before can only be implemented if the effort to calculate the SFP distance to obstacles according to (26) 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 (3) and the constraints (4) and (5). Unfortunately, this task still represents an unsolved problem in the optimal control literature (see, e.g., [21]). A possible remedy is to use the SFP distance induced by removing the restriction (5), i.e., allowing arbitrarily fast steering by setting $\sigma_{\max} \to \infty$. Equivalently, the former state κ can be directly taken as input such that the model in question reduces to

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \end{bmatrix} v \tag{28}$$

with the remaining input constraint $|\kappa| \leq \kappa_{\max}$ [see (4)], the reduced configuration vector $q_r = [x, y, \theta]^T$ and the corresponding reduced configuration space $\mathcal{C}_r$. Obviously, the simplification does not compromise the flatness property (see Section II-D) such that the system (28) is also flat with the flat output (13). Assuming the car model (28) with the constraint (4), Reeds and Shepp [22] 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 $\mathcal{W}$, which result from piecewise constant inputs κ and v. Furthermore, they proved that the shortest trajectories can be classified in 48 three-parameter families, which can be further divided into nine groups. These groups are listed in Table I, where the notation C means an arc of a circle, S denotes a straight-line segment, the separator $|$ indicates a shift in direction and the indexed parameters determine the arc-length of the corresponding trajectory segment. In principle, the calculation of the SFP distance with respect to the simplified model (28) can now be accomplished as follows. Given two configurations $q_{r,1}$ and $q_{r,2}$, 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 (see [20] for details). Note that being able to compute the distance between two single configurations, the evaluation of the obstacle distance according to (26) is also feasible but computationally expensive. However, as will become clear in Section IV-A most of the expensive calculations can be accomplished offline. Furthermore, in order to reduce the error involved by neglecting the restriction (5) for the calculation of the SFP distance, a stricter constraint $|\kappa| \leq \kappa_{\max}^{\mathrm{SFP}} < \kappa_{\max}$ may be used instead of (4) (see [23] for details).

TABLE I
FAMILIES OF SHORTEST TRAJECTORIES FOR (28)

Group	Notation
I	$C_a \vert C_b \vert C_e$
II	$C_a \vert C_b C_e$
III	$C_a C_b \vert C_e$
IV	$C_a C_b \vert C_b C_e$
V	$C_a \vert C_b C_b \vert C_e$
VI	$C_a \vert C_{\frac{\pi}{2\kappa_{max}}} S_l C_{\frac{\pi}{2\kappa_{max}}} \vert C_b$
VII	$C_a \vert C_{\frac{\pi}{2\kappa_{max}}} S_l C_b$
$VIII$	$C_b C_{\frac{\pi}{2\kappa_{max}}} S_l \vert C_a$
IX	$C_a S_l C_b$

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

A small parking duration T as required in the qualitative condition 3) 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* $\mathrm{Steer}_{\mathrm{cc}}$ proposed in [16]. By construction, this particular algorithm yields short trajectories, although no strict optimality condition is satisfied. Moreover, it fulfills the state and the input restrictions (4) and (5) explicitly and can easily be extended to satisfy the topological property (see Section III-C). In the following, the main ideas and properties of the planner are stated (for details see [16] and [24]).

In principle, the CC trajectory planner is a generalization of the shortest trajectories for the simplified system (28) consisting of arcs of circles and straight-line segments in $\mathcal{W}$ (see Table I) to feasible trajectories for the entire system (3). The reason that these trajectories cannot be applied directly to (3)

are their nonsufficient continuity properties. As indicated by the expression (17) for the flatness-based feedforward control the planned reference trajectory for the flat output (20) must at least be three times differentiable with respect to time. However, it is easily verified that for instance a direct transition between straight and circular motion at a constant speed v means an abrupt change in acceleration $[\ddot{y}_{f1,d}, \ddot{y}_{f2,d}]^T$, i.e., a discontinuity in the second-order derivative of $y_{f,d}$. The CC trajectory planner solves this problem by replacing the circular arcs by a more complex structure called *CC-Turn*.

The definition of the CC-Turn is motivated by the symbolic solution of the system differential equations (9) when assuming piecewise constant (bang-bang) inputs v and σ, which, due to the chained-form like structure, can be computed by successive integration. Recall that in terms of the considered parking problem, the arc-length based differential equations (9) are equivalent to the usual time-based model description (3) (see Section II-B). Two basic cases of constant inputs can be distinguished. First, a vanishing input $\sigma = \bar{\sigma} = 0$ which in combination with a constant velocity $v = v_0 \neq 0$ and the arbitrarily chosen start configuration $q(s=0) = [0,0,0,\kappa_0]^T$ results in an arc of a circle in $\mathcal{W}$ with radius $1/\kappa_0$. Second, assuming $v = v_0 \neq 0$ and a nonvanishing constant input $\bar{\sigma} = \bar{\sigma}_0$ which based on the for computational convenience chosen start configuration $q(s=0) = [0,0,0,0]^T$ yields

$$y_{f1}(s) = x(s) = \operatorname{sign}(v_0)\sqrt{\frac{\pi}{|\bar{\sigma}_0|}}\mathrm{F_c}\left(\frac{|\bar{\sigma}_0|}{\pi}s\right) \tag{29}$$

$$y_{f2}(s) = y(s) = \operatorname{sign}(v_0)\operatorname{sign}(\bar{\sigma}_0)\sqrt{\frac{\pi}{|\bar{\sigma}_0|}}\mathrm{F_s}\left(\frac{|\bar{\sigma}_0|}{\pi}s\right) \tag{30}$$

with the Fresnel integrals

$$\mathrm{F_c}(x) = \int_0^x \cos\left(\frac{\pi}{2}\xi^2\right)d\xi, \quad \mathrm{F_s}(x) = \int_0^x \sin\left(\frac{\pi}{2}\xi^2\right)d\xi. \tag{31}$$

Equations (29)–(30) describe a clothoid arc of sharpness $\bar{\sigma}_0$ in $\mathcal{W}$. A CC-Turn is a combination of at most three of these elementary trajectory segments, i.e., circular and clothoid arcs in $\mathcal{W}$, establishing a feasible connecting trajectory between two configurations with $\kappa = 0$ located on a particular so-called *CC-circle* as shown in Fig. 5. The radius r_{cc} of the CC-circle as well as the angle μ between the tangent and the connectable car orientations at any point on the CC-circle are characteristic constants which only depend on the velocity $v_{\max}$ and the value of the constraints $\kappa_{\max}$ and $\sigma_{\max}$. In [16] and [24], it is shown by geometrical arguments that by replacing all circular arcs by CC-Turns the shortest trajectory families for the simplified model, as listed in Table I, can be converted into feasible trajectories for the entire model (3). Employing this approach the CC trajectory planner calculates the arc-length of all 48 trajectories in Table I which exist for a given pair of configurations and returns the shortest one.

However, the procedure must be extended by another trajectory family to ensure the required topological property (see Section III-C). The reason is that the trajectory segment corresponding to a single CC-Turn has a nonvanishing minimum length even if the start and goal configuration are located very close to each other (see [16]). Thus, the unextended CC trajectory planner cannot meet the topological property, since, therefore, the length of the resulting trajectory must tend to zero if the configurations which are to be connected approach each other (see [10]). The remedy as proposed in [16] is the additional definition of a so-called *topological trajectory family* which meets the topological property by construction. However, these *topological trajectories* contain many shifts in direction so that they should be avoided whenever possible (see Section IV-C). Nevertheless, adding the topological trajectory family guarantees the completeness of the overall algorithm.

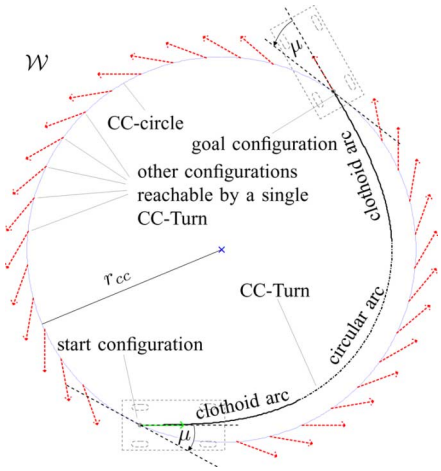


Fig. 5. Example for a CC-Turn

IV. Adaption to the Parking Problem

Having introduced the basic idea and fundamentals of the steering method, the two-step approach must be adapted to meet the specific requirements of the parking problem. In contrast to the preceding sections, where mainly known results from literature have been reviewed and combined to obtain a suitable planning scheme, this task requires the use of some interesting new concepts, which are explained in the following.

A. Obstacle Distribution

As explained in Section III-D, the actual evaluation of the SFP obstacle distance d_{SFP} is very computationally 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 the 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 (26)]. Fig. 6 illustrates how typical parking situations can be approximated using these simple geometrical components. Assuming simplified geometrical 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 q_r [see (28)] on a sufficient narrow grid of sampling points within a

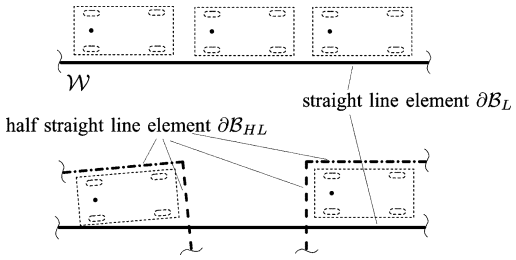


Fig. 6. Approximation of obstacle distributions using straight line ($\partial\mathcal{B}_L$) and half straight line ($\partial\mathcal{B}_{HL}$) boundary sets.

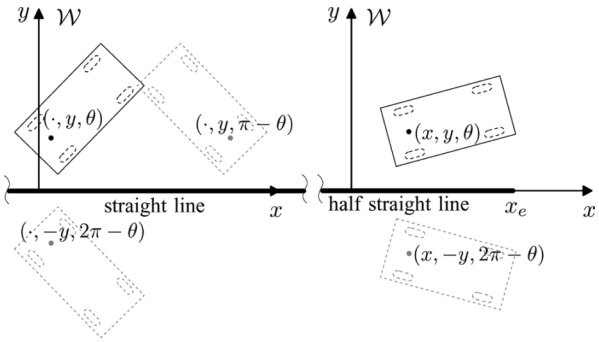


Fig. 7. Symmetry properties of the basic obstacle elements.

significant range (see Section V-A for details). 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 Fig. 6). 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.

It is obvious that the reduction of online computing time faces considerably higher memory requirements since without further measures two 3-D look-up tables must be stored. Recall that the dimension of the configuration space $\mathcal{C}_r$ corresponding to (28) is 3. 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-hand side of Fig. 7. It is immediately apparent that the corresponding SFP distance $d^L_{\mathrm{SFP}}(x, y, \theta) = d_{\mathrm{SFP}}((x, y, \theta)^T, \partial\mathcal{CB}_L)$ does not depend on x meaning that a 2-D look-up table is sufficient. Furthermore, the relations

$$d^L_{\mathrm{SFP}}(x, y, \theta) = d^L_{\mathrm{SFP}}(x, y, \pi - \theta) \tag{32}$$

and

$$d^L_{\mathrm{SFP}}(x, y, \theta) = d^L_{\mathrm{SFP}}(x, -y, 2\pi - \theta) \tag{33}$$

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-hand side of Fig. 7. Obviously, the SFP distance $d^{HL}_{\mathrm{SFP}}(x, y, \theta) = d_{\mathrm{SFP}}((x, y, \theta)^T, \partial\mathcal{CB}_{HL})$ satisfies

$$d^{HL}_{\mathrm{SFP}}(x, y, \theta) = d^{HL}_{\mathrm{SFP}}(x, -y, 2\pi - \theta) \tag{34}$$

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

$$d^{HL}_{\mathrm{SFP}}(x, y, \theta) = d^L_{\mathrm{SFP}}(x, y, \theta) \text{ for } x \ll x_e \tag{35}$$

permits the truncation of small x values in the look-up table corresponding to $\partial\mathcal{B}_{HL}$. Further details of the implementation of the look-up tables are explained in Section V-A.

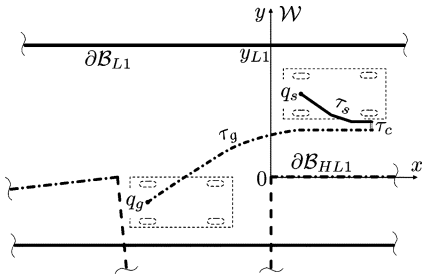


Fig. 8. Algorithm for distance optimized path planning.

B. Distance Optimized Path Planning

In this section, the algorithm used for the online computation of the distance optimized path is explained in detail considering the parallel parking task shown in Fig. 8 as an example. As illustrated, a reference coordinate system is arbitrarily defined by the upper boundary $\partial\mathcal{B}_{HL1}$ of the bottom-right obstacle. Since the algorithm to be presented takes advantage of symmetry properties, it requires that the top straight line obstacle boundary $\partial\mathcal{B}_{L1}$ is parallel to $\partial\mathcal{B}_{HL1}$. Assuming normal parking situations, this restriction usually is satisfied or can at least be enforced by approximating the real upper obstacle environment tolerantly. As outlined in Section III-D, two subpaths τ_s and τ_g starting from q_s and q_g (see Fig. 8) are planned in the 3-D configuration space $\mathcal{C}_r$ at the first step of the algorithm. For this task a discretization of $\mathcal{C}_r$ is applied using the same grid size as for the calculation of the obstacle distance look-up tables. Sequences of discrete successive points $\tau_s^1, \tau_s^2, \ldots, \tau_s^N$ and $\tau_g^1, \tau_g^2, \ldots, \tau_g^M$ on this grid approximating τ_s and τ_g are then determined by the following procedure:

1) set $\tau^1_{(\cdot)} = q_{(\cdot)}$, $k = 0$;
2) set $k \rightarrow k + 1$;
3) define $\tau^{k+1}_{(\cdot)}$ as a nearby point of $\tau^k_{(\cdot)}$ on the discretization grid in $\mathcal{C}_r$ which satisfies the following conditions:
   a) $\tau^{k+1}_{(\cdot),x} \geq \tau^k_{(\cdot),x}$, where $\tau^i_{(\cdot),x}$ denotes the x-component of the configuration $\tau^i_{(\cdot)}$;
   b) $\tau^{k+1}_{(\cdot)} \neq \tau^i_{(\cdot)}, \forall i \leq k$, i.e., avoid multiple occurences of the same configuration;
   c) $d_{\mathrm{SFP}}(\tau^{k+1}_{(\cdot)}, \mathcal{CB}) \geq d_{\mathrm{SFP}}(q^{k+1}, \mathcal{CB})$ for all nearby configurations q^{k+1} of $\tau^k_{(\cdot)}$ being feasible according to the conditions a) and b);

4) if $\tau_{(\cdot),y}^{k+1} = \tau_{(\cdot),y}^{k}$, $\tau_{(\cdot),\theta}^{k+1} = \tau_{(\cdot),\theta}^{k}$ and $\tau_{(\cdot),x}^{k+1} > 0$ then the algorithm is finished, else continue with Step 3) ($\tau_{(\cdot),y}^{i}$ and $\tau_{(\cdot),\theta}^{i}$ mean the y- and the θ-component of $\tau_{(\cdot)}^{i}$, respectively).

The conditions formulated in Step 3) ensure that τ_s and τ_g run in the direction of a local maximum with respect to the obstacle distance. Moreover, having reached a local maximum it is tracked along increasing x-values. Due to the assumed symmetry of the obstacles boundaries $\partial\mathcal{B}_{L1}$ and $\partial\mathcal{B}_{HL1}$ the computed subpaths will always pass a point where the obstacle distance values become independent of x (see the symmetry properties of $\mathcal{B}_L$ and $\mathcal{B}_{HL}$ derived in Section IV-A). The termination condition in Step 4) stops the algorithm as soon as this point is reached.

Having determined the fragments τ_s and τ_g, the remaining task is to connect them to an overall path τ. At first, the subpath ending at a smaller x-value is lengthened by continuing with the previous algorithm until the x-coordinate of the endpoints of τ_s and τ_g coincide. Now, two cases can occur. First, there exists at least one intersection point $q_{is} = \tau_s^{N_{is}} = \tau_g^{M_{is}}$ between τ_s and τ_g with the x-component satisfying $q_{is,x} > 0$ and, where the length of the sequences τ_s and τ_g until intersection is given by N_{is} and M_{is}, respectively. Then, the sampling points τ^i of the overall distance optimized path τ can easily be constructed by following τ_s from q_s to the intersection point q_{is} and τ_g backward from the intersection point to q_g, i.e.,

$$\tau^i = \begin{cases} \tau_s^i, & 1 \le i \le N_{is} \\ \tau_g^{M_{is}+N_{is}-i}, & N_{is} < i \le N_{is} + M_{is} - 1. \end{cases} \tag{36}$$

In the second case, where τ_s and τ_g do not intersect a suitable third subpath fragment τ_c connecting the endpoints of τ_s and τ_g has to be determined (see Fig. 8), i.e., the sequence of the sampling points of τ is defined as

$$\tau^i = \begin{cases} \tau_s^i, & 1 \le i \le N-1 \\ \tau_c^{i-(N-1)}, & N-1 < i \le N-1+L \\ \tau_g^{M+N+L-1-i}, & N+L \le i \le N+L+M-2 \end{cases} \tag{37}$$

where N and M denote the length of the sampling sequences of τ_s and τ_g, respectively, and τ_c^i, $i = 1, 2, \ldots, L$, are the sampling points of τ_c with $\tau_c^1 = \tau_s^N$ and $\tau_c^L = \tau_g^M$. Naturally, τ_c should also avoid all obstacles with the largest possible distance. In order to compute the samples τ_c^i with this property, an analytic expression for the SFP obstacle distance $d_{\mathrm{SFP}}(q_r, \mathcal{CB})$ being valid for all $q_r = [q_{r,x}, q_{r,y}, q_{r,\theta}]^T$ within the significant region of $\mathcal{C}_r$ close to the endpoints of τ_s and τ_g is derived. This is feasible due to the symmetry properties of the obstacle distribution and under the following assumptions which are automatically satisfied by construction of the planning algorithm for τ_s and τ_g. The first condition is that replacing the lower boundary $\partial\mathcal{B}_{HL1}$ (see Fig. 8) by a straight line element $\partial\tilde{\mathcal{B}}_{L1}$ as done in Fig. 9 does not affect the distance calculations. This is fulfilled because the termination condition in Step 4) does not allow τ_s or τ_g to end before that point is reached [see also (35)]. Furthermore, the region of interest determined by the endpoints of τ_s and τ_g has to be located roughly midway between both obstacle boundaries $\partial\mathcal{B}_{L1}$ and $\partial\tilde{\mathcal{B}}_{L1}$ in $\mathcal{W}$ (see Fig. 9) and the corresponding absolute angle values $|\theta|$ must be small. Both requirements are ensured by the (local) maximization of the obstacle distance formulated within Step 3). Finally, the (Euclidean) distance y_{L1} between the top and the bottom obstacle boundaries $\partial\mathcal{B}_{L1}$ and $\partial\tilde{\mathcal{B}}_{L1}$ in $\mathcal{W}$ must not be too large. More precisely, the relation

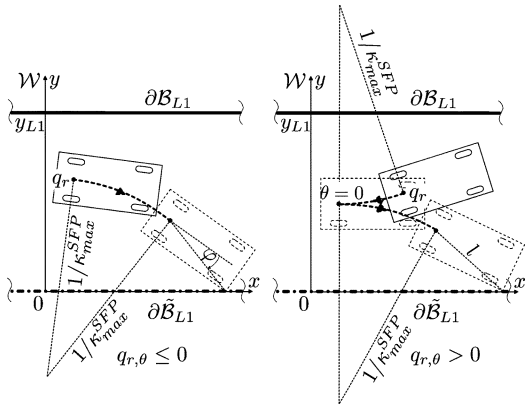


Fig. 9. Derivation of the SFP obstacle distance in a region around the endpoints of τ_s and τ_g; two cases are distinguished: $q_{r,\theta} \le 0$ (left-hand side) and $q_{r,\theta} > 0$ (right-hand side).

$$y_{L1} \ll 2\left(\frac{1}{\kappa_{\max}^{\mathrm{SFP}}} + l\right) \tag{38}$$

should hold, where $\kappa_{\max}^{\mathrm{SFP}}$ is the adjusted upper bound on κ applied for the SFP distance calculation (see Section III-D) and the length l is a car parameter defined in Fig. 9. This last condition is also satisfied in practice since the value on the right-hand side of the inequality (38) becomes large for realistic car parameters. Under these assumptions the shortest trajectories with respect to (28) and the constraint $\kappa \le \kappa_{\max}^{\mathrm{SFP}}$ leading from q_r to a collision with the obstacle boundary $\partial\tilde{\mathcal{B}}_{L1}$ are of the simple forms shown in Fig. 9 which follows from straightforward geometrical arguments. They either consist of a forward circular arc of minimum radius $1/\kappa_{\max}^{\mathrm{SFP}}$ for $q_{r,\theta} \le 0$ or a combination of a backward and a forward circular arc of the same radius for $q_{r,\theta} > 0$. Now, recall that the length of the shortest trajectory in $\mathcal{W}$ leading to a collision with an obstacle is exactly the definition of the SFP obstacle distance (see Section III-D). Thus, determining the arc-length of those elementary trajectories using straightforward geometrical relationships yields the analytic expression for the SFP obstacle distance to $\partial\tilde{\mathcal{B}}_{L1}$

$$d_{\mathrm{SFP}}(q_r, \partial\mathcal{C}\tilde{\mathcal{B}}_{L1}) = \begin{cases} \frac{1}{\kappa_{\max}^{\mathrm{SFP}}}\left(q_{r,\theta} + \arcsin\left[R\left(q_{r,y} - \frac{1}{\kappa_{\max}^{\mathrm{SFP}}}\right)\right] + \vartheta\right), & q_{r,\theta} > 0 \\ \frac{1}{\kappa_{\max}^{\mathrm{SFP}}}\left(q_{r,\theta} + \arcsin\left[R\left(q_{r,y} - \frac{\cos q_{r,\theta}}{\kappa_{\max}^{\mathrm{SFP}}}\right)\right] + \vartheta\right), & q_{r,\theta} \le 0 \end{cases} \tag{39}$$

where the positive constants

$$R = \kappa_{\max}^{\mathrm{SFP}} / \sqrt{(\kappa_{\max}^{\mathrm{SFP}} l)^2 - 2\kappa_{\max}^{\mathrm{SFP}} l \sin\varphi + 1}$$

and

$$\vartheta = (\pi/2) - \arctan(\kappa_{\max}^{\mathrm{SFP}} l \cos\varphi / (1 - \kappa_{\max}^{\mathrm{SFP}} l \sin\varphi))$$

are used and $\partial\mathcal{C}\tilde{\mathcal{B}}_{L1}$ means the boundary set of the configuration obstacle corresponding to $\partial\tilde{\mathcal{B}}_{L1}$. The SFP distance with respect to the top obstacle

$$d_{\mathrm{SFP}}(q_r, \partial\mathcal{C}\mathcal{B}_{L1}) = \begin{cases} \frac{1}{\kappa_{\max}^{\mathrm{SFP}}}\left(-q_{r,\theta} + \arcsin\left[R\left(y_{L1} - q_{r,y} - \frac{1}{\kappa_{\max}^{\mathrm{SFP}}}\right)\right] + \vartheta\right), & q_{r,\theta} < 0 \\ \frac{1}{\kappa_{\max}^{\mathrm{SFP}}}\left(-q_{r,\theta} + \arcsin\left[R\left(y_{L1} - q_{r,y} - \frac{\cos q_{r,\theta}}{\kappa_{\max}^{\mathrm{SFP}}}\right)\right] + \vartheta\right), & q_{r,\theta} \geq 0 \end{cases} \tag{40}$$

follows from (39) by symmetry giving rise to the required analytical expression

$$d_{\mathrm{SFP}}(q_r, \mathcal{C}\mathcal{B}) = \min\left(d_{\mathrm{SFP}}(q_r, \partial\mathcal{C}\tilde{\mathcal{B}}_{L1}), d_{\mathrm{SFP}}(q_r, \partial\mathcal{C}\mathcal{B}_{L1})\right) \tag{41}$$

for the SFP distance to the entire obstacle distribution. In view of (41), a suitable connecting subpath τ_c can be constructed. Since it is immediately checked that (39) strictly increases whereas (40) strictly decreases with respect to $q_{r,y}$ and $q_{r,\theta}$ within the significant region of $\mathcal{C}_r$, local maxima of (41) can only occur on the line of intersection, where

$$d_{\mathrm{SFP}}(q_r, \partial\mathcal{C}\tilde{\mathcal{B}}_{L1}) = d_{\mathrm{SFP}}(q_r, \partial\mathcal{C}\mathcal{B}_{L1}). \tag{42}$$

Thus, the endpoints of the subpaths τ_s and τ_g will always be located on this line of intersection, which can be described explicitly shown by (43) at the bottom of the page, following from (42) by applying basic simplifications. Moreover, defining τ_c by moving along (43) assures the maximum possible SFP obstacle distance. The sampling points of τ_c needed for the actual calculation of τ according to (37) can easily be calculated in view of the explicit form of (43).

C. Algorithm for Trajectory Generation

Having determined a discrete representation of a suitable collision free path τ, the second step is the transformation of τ into an admissible trajectory, i.e., a trajectory that can be exactly tracked by (3) being subject to the restrictions (4) and (5). In the sequel, a possible algorithm accomplishing that task will be presented in detail, which can be used to implement the basic procedure presented in Section III-B.

First, recall the planning scheme outlined in Section III-B, where a sequence of sampling points $\bar{q}_i$ [see (21)] on the collision-free path τ are chosen and connected pairwise using a local trajectory planner. Naturally, one could think of many different ways to generate a more or less appropriate sequence of sampling points. However, as becomes obvious when considering the properties of the CC trajectory planner (see Section III-E) the choice of the sequence of sampling points $\bar{q} = \{\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_p\}$ on τ is crucial for the performance of the overall approach. Especially close-by or unnecessarily many sampling points will lead to lengthy and complex trajectories containing many shifts in direction. Having tested several different procedures, it turned out that an algorithm, where the sequence $\bar{q}$ and the corresponding trajectory are constructed iteratively by moving backward from q_g to q_s, yields good results. To state this more precisely, suppose that a discrete representation of a collision-free path τ is given by the sequence of points τ^i, $i = 1, \ldots, P$, with $\tau^1 = q_s$ and $\tau^P = q_g$. Furthermore, introduce the sequence of indices $\bar{\eta}$ as

$$\bar{\eta} = \{\bar{\eta}_1, \bar{\eta}_2, \ldots, \bar{\eta}_p\} \tag{44}$$

such that if the $\bar{\eta}_i$th sampling point $\tau^{\bar{\eta}_i}$ of the collision-free path is chosen as the ith sampling point $\bar{q}_i$ this can be represented by

$$\bar{q}_i = \tau^{\bar{\eta}_i}, \quad i = 1, 2, \ldots, p. \tag{45}$$

Now, an overall collision-free trajectory between q_s and q_g can be constructed iteratively by proceeding as follows.

1) Initialize the sequence of sampling points as $\bar{q} = \{\bar{q}_1\}$ with $\bar{q}_1 = \tau^P = q_g$ yielding the corresponding sequence of indices $\bar{\eta} = \{\bar{\eta}_1\}$ with $\bar{\eta}_1 = P$.

2) If $\mathrm{Steer}_{\mathrm{cc}}(q_s, \bar{q}_1, t)$ is not collision-free

THEN

2a1) Set $\hat{\eta} = \lfloor (1 + \bar{\eta}_1)/2 \rfloor$, where $\lfloor \xi \rfloor$, $\xi \in \mathbb{R}$, means the nearest integer less than or equal to ξ

2a2) If $\mathrm{Steer}_{\mathrm{cc}}(\tau^{\hat{\eta}}, \bar{q}_1, t)$ is not collision-free

THEN

— Set $\hat{\eta} \rightarrow \hat{\eta} + 1$

— Continue with Step 2a2)

ELSE

— Add $\mathrm{Steer}_{\mathrm{cc}}(\tau^{\hat{\eta}}, \bar{q}_1, t)$ to the overall trajectory

— Set $\bar{q} \rightarrow \{\tau^{\hat{\eta}}, \bar{q}\}$ and $\bar{\eta} \rightarrow \{\hat{\eta}, \bar{\eta}\}$

(Note that consequently $\bar{q}_1 \rightarrow \tau^{\hat{\eta}}$ and $\bar{\eta}_1 \rightarrow \hat{\eta}$)

— Continue with Step 2)

$$q_{r,y} = \begin{cases} \frac{1}{2\kappa_{\max}^{\mathrm{SFP}}}\left(1 - \cos q_{r,\theta} + \kappa_{\max}^{\mathrm{SFP}} y_{L1} - \tan q_{r,\theta}\sqrt{\left(\frac{2\kappa_{\max}^{\mathrm{SFP}} \cos q_{r,\theta}}{R}\right)^2 - \left(y_{L1}\kappa_{\max}^{\mathrm{SFP}} - 1 - \cos q_{r,\theta}\right)^2}\right), & q_{r,\theta} > 0 \\ \frac{1}{2\kappa_{\max}^{\mathrm{SFP}}}\left(\cos q_{r,\theta} - 1 + \kappa_{\max}^{\mathrm{SFP}} y_{L1} - \tan q_{r,\theta}\sqrt{\left(\frac{2\kappa_{\max}^{\mathrm{SFP}} \cos q_{r,\theta}}{R}\right)^2 - \left(y_{L1}\kappa_{\max}^{\mathrm{SFP}} - 1 - \cos q_{r,\theta}\right)^2}\right), & q_{r,\theta} \leq 0 \end{cases} \tag{43}$$

ELSE

2b1) Add $\mathrm{Steer}_{\mathrm{cc}}(q_s, \bar{q}_1, t)$ to the overall trajectory

2b2) Set $\bar{q} \rightarrow \{q_s, \bar{q}\}$ and $\bar{\eta} \rightarrow \{1, \bar{\eta}\}$

2b3) Finish the algorithm.

Note that testing a subtrajectory for collisions with obstacles as required in Steps 2) and 2a2) can be approximately accomplished in a computationally efficient way by applying the existent distance look-up tables used for path planning (see Section IV-A). To this end, discrete points of the subtrajectory are determined and their corresponding SFP distances to all obstacles are looked-up in memory. A subtrajectory is declared collision-free, if all distance values are greater than zero or, alternatively, greater than a given minimum safety distance. It should also be mentioned that a problem occurs if the available free space is so limited that the nearest points on τ with respect to the chosen discretization grid still cannot be connected by a collision-free trajectory. In this case, adding-up of $\hat{\eta}$ within Step 2a2) can reach a point where $\hat{\eta} = \bar{\eta}_1$. One way to handle this situation is to introduce another sampling point $\hat{\tau}$ on τ between $\tau^{\bar{\eta}_1}$ and $\tau^{\bar{\eta}_1 - 1}$, e.g., by setting $\hat{\tau} = (1/2)(\tau^{\bar{\eta}_1 - 1} + \tau^{\bar{\eta}_1})$. This can be done repeatedly until a collision-free subtrajectory between $\hat{\tau}$ and $\tau^{\bar{\eta}_1}$ is found, which is guaranteed by the topological property of $\mathrm{Steer}_{\mathrm{cc}}$.

D. Automatic Adaption of the Maximum Velocity

Although the CC trajectory planner fulfills the topological property for arbitrary $v_{\max} > 0$ [see (7)], meaning that a solution will always be found, the resulting trajectories will not satisfy the qualitative condition 4) of the parking problem, if the parking space is small and $v_{\max}$ is large. This coincides with the intuition that high-speed maneuvering is not favorable inside of narrow parking spaces. On the other hand, setting $v_{\max}$ to a small value right from the start contradicts the requirement for a small parking duration (see condition 3) of the parking problem), since this choice also slows down the noncritical maneuvers outside of the parking spot. To overcome these difficulties, an automatic adaption procedure for $v_{\max}$ is proposed which makes use of the structure of the planning algorithm presented before. The basic idea is that the maximum allowed velocity can be adjusted while scanning for the next suitable sampling point on τ [see Step 2a2)], e.g., each resulting subtrajectory ends up with its own associated maximum velocity value $v_{\max}^{\mathrm{adapt}}$. The distance between the intermediate goal configuration $\bar{q}_1$ and the current candidate sampling point $\tau^{\hat{\eta}}$ is taken as criterion to determine a suitable value for $v_{\max}^{\mathrm{adapt}}$. The idea is that, if the sampling points to be connected approach each other, then only few free space is available since otherwise a collision-free trajectory would have been found before. Furthermore, as already has been mentioned in Section III-E, the length of all "normal" trajectories calculated by $\mathrm{Steer}_{\mathrm{cc}}$ is lower bounded such that the connection of close configurations automatically leads to the topological trajectory family, which contains many maneuvers by construction (see Section III-E). Motivated by this last aspect and the fact that the minimum "normal" trajectory length $l_{\min}$ depends on the maximum allowed velocity the condition

$$K l_{\min}\left(v_{\max}^{\mathrm{adapt}}\right) \leq d_{\mathrm{SFP}}(\bar{q}_1, \tau^{\hat{\eta}}) \tag{46}$$

for determining $v_{\max}^{\mathrm{adapt}}$ can be formulated, where $d_{\mathrm{SFP}}(\cdot,\cdot)$ means the shortest feasible path metric as defined in (24) and the constant parameter $K > 0$ can be used to tune the adaption algorithm. According to [16], the minimum trajectory length $l_{\min}$ is given by

$$l_{\min}(v_{\max}) = 4\left[\sqrt{\frac{\pi v_{\max}}{\sigma_{\max}}} F_c\left(\frac{\kappa_{\max}^2 v_{\max}}{\pi \sigma_{\max}}\right) - \frac{1}{\kappa_{\max}} \sin\frac{\kappa_{\max}^2 v_{\max}}{2\sigma_{\max}}\right] \tag{47}$$

with $F_c(\cdot)$ denoting the Fresnel cosine integral function [see (31)]. For small velocities $v_{\max}$ and realistic car parameters $\kappa_{\max}$ and $\sigma_{\max}$ the first-order Taylor series approximation

$$l_{\min}(v_{\max}) \approx 2\frac{\kappa_{\max}}{\sigma_{\max}} v_{\max} \tag{48}$$

represents a very good estimate for the complex relationship (47). Thus, inserting (48) into (46) yields the expression

$$v_{\max}^{\mathrm{adapt}} = \frac{\sigma_{\max}}{2K\kappa_{\max}} d_{\mathrm{SFP}}(\bar{q}_1, \tau^{\hat{\eta}}) \tag{49}$$

which can be used to adapt the allowed velocity range at each iteration Step 2a2) of the trajectory planning algorithm. As discussed in Section III-D, the evaluation of the exact distance $d_{\mathrm{SFP}}(\cdot,\cdot)$ with respect to the entire car model (3) and the constraints (4) and (5) is difficult such that the corresponding distance measure with respect to the simplified system (28) may be used instead in order to evaluate (49). Obviously, the actual value of $v_{\max}^{\mathrm{adapt}}$ should be restricted to its nominal value, i.e.,

$$v_{\max}^{\mathrm{adapt}} \leq v_{\max}. \tag{50}$$

Since the actual velocity $v(t)$ is determined by the driver (see Section II-B), who consequently has to observe its upper limit by himself, very small values of $v_{\max}^{\mathrm{adapt}}$ are not practicable for the real application. Thus, $v_{\max}^{\mathrm{adapt}}$ should also be restricted to satisfy the lower bound

$$v_{\max}^{\mathrm{adapt}} \geq v_{\max}^{low} \tag{51}$$

with a suitable lower velocity limit $v_{\max}^{low} < v_{\max}$. However, if the parking space is very small such that (51) has to be applied, then without further measures the use of the unsuitable topological trajectory family becomes likely again. In that case, a better alternative might be to allow steering at standstill. This can be realized by switching to another local trajectory planner $\overline{\mathrm{Steer}}$ calculating the shortest trajectories for the simplified model (28) (see Section III-D and Table I) as soon as $v_{\max}^{\mathrm{adapt}}$ according to

(49) violates the condition (51). Fortunately, the application of this idea does not involve the implementation of a completely new local trajectory planner $\overline{\mathrm{Steer}}$. As shown in [16], the trajectories evaluated by $\mathrm{Steer}_{\mathrm{cc}}$ converge to the shortest trajectories of the model (28) for $\sigma_{\max} \to \infty$. Thus, in the implementation switching to $\overline{\mathrm{Steer}}$ can be realized by simply setting $\sigma_{\max}$ to a very large numerical value.

V. Implementation Issues

This section focuses on two particular implementation aspects. First, the computation of the obstacle distance look-up tables used for path planning (see Section IV-A) and collision detection (see Section IV-C) is discussed in detail. Second, some simple but very important and effective measures to reduce the online computation time are briefly described.

A. Calculation of Obstacle Distance Look-Up Tables

The principle of a numerical algorithm for the actual computation of the entries of the SFP obstacle distance look-up table is explained in the following. Furthermore, an approach for choosing a suitable discretization grid is discussed.

Assuming that an algorithm for the SFP distance evaluation $d_{\mathrm{SFP}}(q_1, q_2)$ between two discrete configurations q_1 and q_2 is available (see Section III-D), the calculation of the SFP obstacle distance $d_{\mathrm{SFP}}(q, \partial\mathcal{CB}_i)$ between a configuration q and an obstacle $\mathcal{CB}_i$ can be accomplished by means of a direct numerical implementation of expression (26). 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. For the actual realization of the first step, one uses the fact 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 (see [23] for details). The second step for actually evaluating $d_{\mathrm{SFP}}(q, \partial\mathcal{CB}_{HL})$ is straightforward.

Finally, some remarks regarding a suitable choice of the discretization grid are given. It is obvious that a tradeoff between accuracy and memory requirements is necessary, which can only be made, if the specific requirements for the application and the available hardware resources are known. However, having decided about a certain step-width δ in the car position coordinates x and y the step-width $\Delta\theta$ related to the car orientation θ should be chosen accordingly considering the specific properties of the SFP distance metric. One way to obtain step-widths resulting in approximately the same accuracy is to compare the minimum SFP distances of configurations separated by δ and $\Delta\theta$, respectively. The SFP distance of two configurations separated by at least δ in x (or y) is lower bounded by

$$d_{\mathrm{SFP}}\left([x, y, \theta]^T, [x + \delta, \hat{y}, \hat{\theta}]^T\right) \geq \delta \, \forall x, y, \theta, \hat{y}, \hat{\theta} \qquad (52)$$

where equality holds for $\hat{y} = y$ and $\hat{\theta} = \theta = 0$, i.e., if the start and goal configuration lie on the line determined by the car axis.

TABLE II
Car Parameters Used in the Simulations

Parameter	Symbol	Value
Width of the car	-	1.8m
Length of the car	-	4.2m
Wheelbase	L	2.58m
Distance between rear end and rear axle	-	0.74m
Maximum average steering angle	ϕ_{max}	0.64rad
Maximum steering velocity	ω_{max}	0.43rad/s

In contrast, the SFP distance of two configurations separated by the orientation difference $\Delta\theta$ differ at least by

$$d_{\mathrm{SFP}}\left([x, y, \theta]^T, [\hat{x}, \hat{y}, \theta + \Delta\theta]^T\right) \geq \frac{\Delta\theta}{\kappa_{\max}^{\mathrm{SFP}}} \, \forall x, y, \theta, \hat{x}, \hat{y} \quad (53)$$

which is due to the fact that a car modeled by (28) with the input κ restricted to $|\kappa| \leq \kappa_{\max}^{\mathrm{SFP}}$ must perform a circular movement of minimum arc-length $\Delta\theta/\kappa_{\max}^{\mathrm{SFP}}$ for changing its orientation by $\Delta\theta$ (see Sections III-D and IV-A). Motivated by (52) and (53) one should choose $\Delta\theta$ such that

$$\Delta\theta \lessapprox \kappa_{\max}^{\mathrm{SFP}} \delta. \qquad (54)$$

However, due to the complex nature of the SFP distance calculation, no easy general relationship between the step-width of the discretization grid and the resulting accuracy with respect to the SFP obstacle distance can be formulated. Thus, having chosen a certain discretization grid one should always check that the differences in the SFP distance values between two nearby configurations on the grid are not too large.

B. Optimization of the Online Computing Time

In this section, some important measures to reduce the online computing time are very briefly stated in general terms. First, it is important to understand that the computation time is mainly determined by the local trajectory planner $\mathrm{Steer}_{\mathrm{cc}}$ which is repeatedly applied during the trajectory generation step (see Section III-E and IV-C). As indicated in Section III-E, many nontrivial mathematical functions like Fresnel integrals [see (31)] have to be evaluated during the course of the algorithm which can reduce its performance significantly. Fortunately, the arguments of most of the computational expensive functions can be restricted to a limited range such that they can be efficiently implemented using look-up tables. However, the square root operation, which also has to be accomplished quite frequently (see [16]), cannot be handled this way because it has to be accomplished for a wide range of arguments. However, more detailed analysis shows that during one call of the planner the same square root is extracted within many different parts of the algorithm. Thus, the calculation time can be reduced by carefully avoiding multiple evaluations.

VI. Simulation Results

The following simulations are based on the car parameters listed in Table II, which correspond approximately to the configuration of a VW Golf. The maximum steering velocity $\omega_{\max} = 0.43$ rad/s characterizing the steering actuator means that the steering angle can be changed from maximum left to maximum right in about 3 s. Applying the parameters in Table II to (4) and

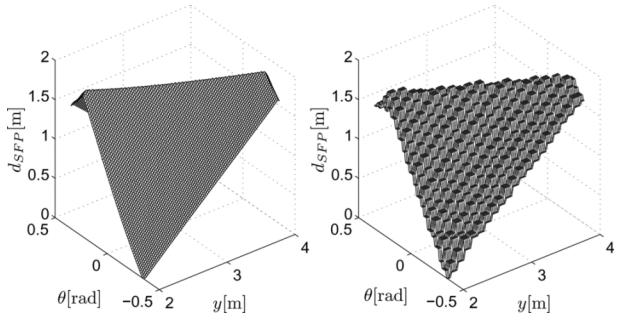


Fig. 10. Accuracy of obstacle distance look-up tables; exact (left-hand side) and approximate d_{SFP}-values (right-hand side) for the obstacle distribution of Fig. 9 with $y_{L1} = 6.0$ m.

(5) leads to the constraints $\kappa_{\max} = 0.291(1/\mathrm{m})$ and $\sigma_{\max} = 0.166(1/\mathrm{ms})$ of the simplified state space model (3). For the calculation of the SFP distances the stricter upper bound $\kappa_{\max}^{\mathrm{SFP}} = 0.223(1/\mathrm{m})$ is assumed (see Section III-D and [23]).

Before the performance of the algorithm is verified, the accuracy, the range, and the memory requirements of the SFP distance look-up tables used in this particular case are discussed. The discretization grid is specified by

$$\delta = 0.1 \text{ m}, \quad \Delta\theta = \frac{1}{90}\pi \tag{55}$$

which roughly satisfy the condition (54). The accuracy being achieved by (55) can be checked by considering the special obstacle configuration in Fig. 9, for which the exact analytical expressions (39) and (40) for the SFP obstacle distance are available. Fig. 10 shows the comparison of the exact and the approximate SFP distance distribution, where the approximation is evaluated by simply using the nearest value stored in the look-up tables. The maximum absolute error is about 0.17 m, which is reasonable in view of the selected discretization grid (55). Considering the symmetry properties of the basic obstacle elements (see Section IV-A) the range of the 2-D straight line distance look-up table is chosen as 0.7 m $\leq y \leq$ 8.0 m, $0 \leq \theta \leq \pi$ and the 3-D half straight line distance look-up table is computed for the region -8.0 m $\leq x \leq$ 8.0 m, 0 m $\leq y \leq$ 8.0 m and $0 \leq \theta \leq 2\pi$, where the endpoint x_e of the half straight line obstacle element is set to $x_e = 0$ (see Fig. 7). Straightforward calculations show that these ranges of the distance look-up tables are sufficient for all obstacle configurations where the top and the bottom straight line boundary (see Fig. 6) are separated by less than $y_{\max} = 11.5$ m (see [23] for details). In view of (55), these figures result in a total of $74 \cdot 91 = 6734$ and $161 \cdot 81 \cdot 180 \approx 2.3 \cdot 10^6$ values to be stored, respectively. Since for the specified $y_{\max}$ the SFP distance values lie in the interval [0 m,4.8 m] with a computational accuracy of less than 0.17 m (see Fig. 10), 1 byte per value ensures a sufficient numerical precision. Thus, adding the memory required by the program code and the resources needed for the optimization of the computing time (see Section V-B), the memory requirements of the overall algorithm amounts to a total of about 3 MB.

The parallel parking situation in Fig. 11 verifies that the algorithm is able to handle tiny parking spaces. In this particular case, the parking space is only 0.9 m longer than the car. Although an unusual start configuration is chosen, the car is transfered to the required goal configuration with only four intermediate halts. As in the following simulation, the initial maximum velocity $v_{\max} = 3$ km/h is assumed and the automatic adaption of $v_{\max}$ to $v_{\max}^{\mathrm{adapt}}$ as explained in Section IV-D is applied. The latter is parametrized by setting $K = 5.0$ [see (46)] and allowing steering at standstill as soon as the threshold $v_{\max}^{\mathrm{low}} = 1$ km/h [see (51)] is undershot. The resulting maximum velocities for each trajectory section stated within the figures show clearly how the mechanism works. Without this measure the resulting trajectory for the situation illustrated in the Fig. 11 would be very complex comprising several hundred changes in the direction of motion.

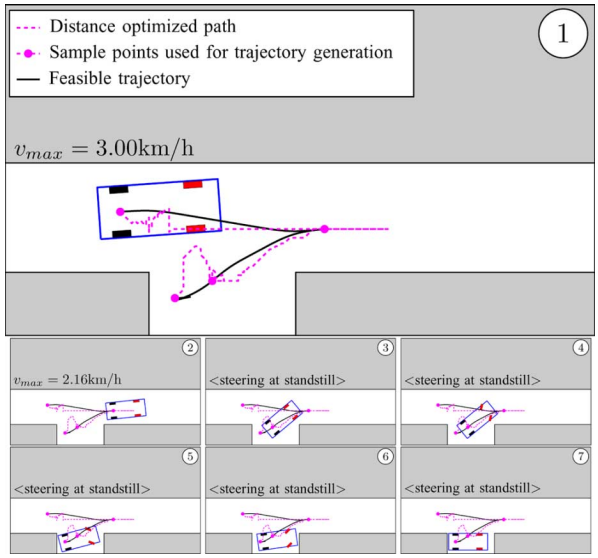


Fig. 11. Distance optimized path and resulting trajectory for a parallel parking space which is 0.9 m longer and 0.4 m wider than the body of the car.

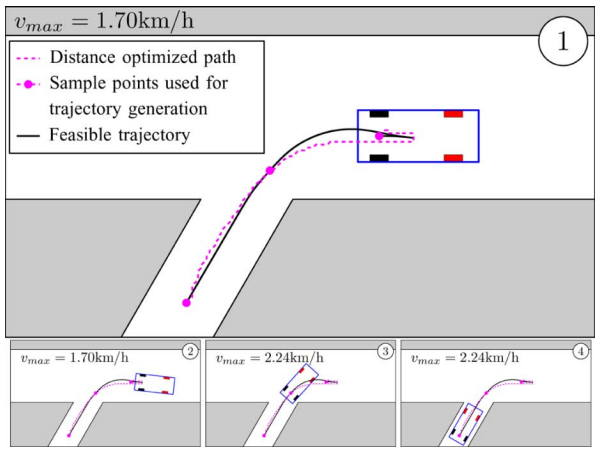


Fig. 12. Distance optimized path and resulting trajectory for a parking space, which is inclined by $\pi/3$ and is 1.0 m wider than the body of the car.

In the last example, an inclined parking space is given in Fig. 12. In principle, the same algorithm as for parallel parking spaces can be applied to handle this situation. However, if the parking space is aligned vertically or inclined by more than $\pi/4$ as in Fig. 12, the algorithm for calculating the distance optimized path should be slightly modified by planning in the direction of positive y-values instead of positive x-values (see Section IV-B). The reason is fairly obvious. Nevertheless, the

algorithm also works without this modification but numerical errors in the distance look-up tables can lead to a path containing several oscillating-like sections in this case.

Using the current implementation in MATLAB, the total computation times for the examples in Figs. 11 and 12 were well beyond 2.5 s on an AMD Athlon XP 1800+. Thus, taking into account that the current software is not optimized with respect to the computation time and that MATLAB is a resources consuming platform, the figures indicate that suitable calculation times for a real application in a car are feasible, if a modern digital signal processor is applied.

VII. Conclusion

In this contribution, a general nonholonomic motion planning scheme in the presence of obstacles was applied to determine trajectories for an autonomous parking maneuver of a car. The requirements of a real life application were met by applying suitable algorithms for the two single steps of the overall planning procedure. The size of the parking space was taken into account by an adaptive choice of the maximum allowed velocity during the parking maneuver. Using a smart obstacle representation to approximate parking environments the online calculation times could be reduced significantly by storing computational expensive data in look-up tables.

References

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

[2] J. Barraquand and J.-C. Latombe, "On non-holonomic mobile robots and optimal maneuvering," *Rev. Intell. Artif.*, vol. 3, no. 2, pp. 77–103, 1989.

[3] A. Divelbiss and J. Wen, "A path space approach to nonholonomic motion planning in the presence of obstacles," *IEEE Trans. Robot. Autom.*, vol. 13, no. 3, pp. 443–451, Jun. 1997.

[4] Y. Chitour and H. Sussmann, "Line-integral estimates and motion planning using a continuation method," in *Essays on Mathematical Robotics*. New York: Springer, 1998.

[5] Y. Chitour, "A continuation method for motion-planning problems," *ESIAM*, vol. 12, pp. 139–168, 2006.

[6] M. Sugeno and K. Murakami, "Fuzzy parking control of a model car," in *Proc. IEEE Conf. Dec. Control*, 1984, pp. 902–903.

[7] F. Cuesta and A. Ollero, *Intelligent Mobile Robot Navigation*. New York: Springer, 2005.

[8] D. Gorinevsky, A. Kapitanovsky, and A. Goldenberg, "Neural network architecture for trajectory generation and control of automated car parking," *IEEE Trans. Control Syst. Technol.*, vol. 4, no. 1, pp. 50–56, Jan. 1996.

[9] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[10] J. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*. New York: Springer, 1998.

[11] I. Kolmanovsky and N. McClamroch, "Developments in nonholonomic control problems," *IEEE Control Syst. Mag.*, vol. 15, no. 6, pp. 20–36, Dec. 1995.

[12] J. Shyu and C. Chuang, "Automatic parking device for automobile," U.S. Patent 4 931 930, Jun. 5, 1990.

[13] I. Paromtchik and C. Laugier, "Motion generation and control for parking an autonomous vehicle," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1996, pp. 3117–3122.

[14] M. Wada, K. Yoon, and H. Hashimoto, "Development of advanced parking assistance system," *IEEE Trans. Ind. Electron.*, vol. 50, no. 1, pp. 4–17, Feb. 2003.

[15] B. Mirtich and J. Canny, "Using skeletons for nonholonomic path planning among obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1992, pp. 2533–2540.

[16] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Trans. Robot. Autom.*, vol. 20, no. 6, pp. 1025–1035, Dec. 2004.

[17] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *Int. J. Control*, vol. 61, pp. 1337–1361, 1995.

[18] P. Rouchon, M. Fliess, J. Lévine, and P. Martin, "Flatness and motion planning: The car with n trailers," in *Proc. Eur. Control Conf.*, 1993, pp. 1518–1522.

[19] J. Laumond, "Feasible trajectories for mobile robots with kinematic and environment constraints," in *Proc. Int. Conf. Intell. Autonomous Syst.*, 1986, pp. 346–354.

[20] J. Laumond and P. Souéres, "Metric induced by the shortest paths for a car-like mobile robot," in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 1993, pp. 1299–1303.

[21] H. Sussmann, "The Markov-Dubins problem with angular acceleration control," in *Proc. 36th IEEE Conf. Dec. Control*, 1997, pp. 2639–2643.

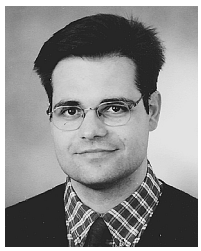
[22] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forward and backward," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.

[23] B. Müller, J. Deutscher, and S. Grodde, "Trajectory design and feedforward control for car parking," Lehrstuhl für Regelungstechnik, Univ. Erlangen-Nürnberg, Erlangen, Germany, 2006.

[24] S. Grodde, "A local trajectory planner for an autonomous parking application," (in German) Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, Erlangen, Germany, 2005.

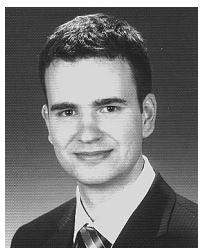
Bernhard Müller received the Dipl.-Ing. Univ. degree in electrical engineering from the Universität Erlangen-Nürnberg, Erlangen, Germany, in 2004.

Since 2004, he has been with the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, as a Research Assistant. His main research interests include nonlinear and automotive control.



Joachim Deutscher received the Dipl.-Ing. (FH) degree in electrical engineering from Fachhochschule Würzburg-Schweinfurt-Aschaffenburg, Schweinfurt, Germany, in 1996, and the Dipl.-Ing. Univ. degree in electrical engineering and the Dr.-Ing. degree from Universität Erlangen-Nürnberg, Erlangen, Germany, in 1999 and 2003, respectively.

Since 1999, he has been with the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, Erlangen, Germany, where he is currently Head of the nonlinear Control Systems Group. His main research interests include nonlinear control and polynomial matrix methods for multivariable control system design.



Stefan Grodde received the Dipl.-Ing. Univ. degree in electrical engineering from the Universität Erlangen-Nürnberg, Erlangen, Germany, in 2005.

Since July 2005, he has been with a well-known supplier of the automotive industry in Stuttgart as a Software Engineer. His main research interests include, in general, automotive control and, in particular, motor management.