Hindawi
Journal of Advanced Transportation
Volume 2017, Article ID 2521638, 27 pages
https://doi.org/10.1155/2017/2521638

WILEY | Hindawi

Research Article

Autonomous Path Planning for Road Vehicles in Narrow Environments: An Efficient Continuous Curvature Approach

Domokos Kiss and Gábor Tevesz

Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Budapest, Hungary

Correspondence should be addressed to Domokos Kiss; domokos.kiss@aut.bme.hu

Received 5 May 2017; Accepted 29 August 2017; Published 31 October 2017

Academic Editor: Mohammad Hamdan

In this paper we introduce a novel method for obtaining good quality paths for autonomous road vehicles (e.g., cars or buses) in narrow environments. There are many traffic situations in urban scenarios where nontrivial maneuvering in narrow places is necessary. Navigating in cluttered parking lots or having to avoid obstacles blocking the way and finding a detour even in narrow streets are challenging, especially if the vehicle has large dimensions like a bus. We present a combined approximation-based approach to solve the path planning problem in such situations. Our approach consists of a global planner which generates a preliminary path consisting of straight and turning-in-place primitives and a local planner which is used to make the preliminary path feasible to car-like vehicles. The approximation methodology is well known in the literature; however, both components proposed in this paper differ from existing similar planning methods. The approximation process with the proposed local planner is proven to be convergent for any preliminary global paths. The resulting path has continuous curvature which renders our method well suited for application on real vehicles. Simulation experiments show that the proposed method outperforms similar approaches in terms of path quality in complicated planning tasks.

1. Introduction

Looking around in the automotive industry, we can see a great amount of automation and an accelerating pursuit to develop intelligent self-driving vehicles. According to the popularity of the field, research is considerably active in the last decades [1–5]. The navigation task of autonomous robotic vehicles consists of many subproblems, involving sensing, perception and recognition of situations, localization, motion planning, and execution. Amongst these, we treat here the problem of generating appropriate paths for robotic cars in the presence of obstacles. As usual in case of cars, we assume the workspace of interest being a planar surface. From a control theoretic view, the rolling without slipping constraint of wheels induces nonholonomic constraints, which do not prohibit the controllability of a system but cause remarkable difficulties (it is enough to think of parallel parking a car). Although controlling a nonholonomic vehicle is a control theoretic challenge, it is worth incorporating knowledge about the specific system in the path planning task as well.

This paper describes a path planning algorithm which is intended to solve navigation problems where the vehicle has to cross narrow or cluttered areas in order to reach the goal. Since cars have a nonzero minimum turning radius, such environments require in many cases nontrivial maneuvering (including more reversals) between narrowing. Our solution is appropriate for the continuous steering car, which means that the finite steering rate is taken into account as well beyond the turning radius. In other words, steering at standstill is avoided by forcing the path to have a continuous curvature profile. This is achieved by using clothoid curves between straight and circular path segments. The proposed algorithm is an approximation-based method consisting of a local and a global planner. The local planner generates feasible paths in the absence of obstacles. The global planner delivers a preliminary collision-free path by omitting the minimum turning radius constraint of the vehicle. It is shown that the proposed local and global planners are appropriate to be applied together and guarantee finding a feasible solution in the presence of obstacles if the planning task is solvable.

Moreover, the local paths are designed to be similar to those a human driver would find.

Although different human drivers and passengers prefer different types of paths, we try to summarize the main characteristics of a "human-like" path as follows. Existing optimal planners strive to return the shortest possible solution which results in sharp (mostly maximal curvature) turns and many reversals in most cases. In our opinion optimal length is not as important for a passenger as feeling comfortable. If a path is longer than optimal (to a meaningful extent, of course) but contains less sharp turns and reversals, then we treat it better from the passenger's perspective. Our proposed approach is designed with these preferences in mind: the preliminary global path consists only of straight segments (with in-place turning between them), and the local planner has the task of connecting these segments with smooth and not necessarily maximal curvature turns.

The remaining part of the paper is organized as follows. The next section summarizes related work in path planning for car-like vehicles. Section 3 introduces the main characteristics of the proposed approach and the previous work of the authors in this field. In Section 4 we describe the local planning component (TTS-planner) in detail, in which we discuss the applied path elements, the properties of local paths, and a topological admissible steering method based on these. In Section 5 the global planning tool (RTR-planner) is described which generates the collision-free preliminary path. Finally, simulation examples are presented which confirm the advantageous characteristics of the proposed approach.

2. Related Work

The path planning task for nonholonomic systems is difficult, caused by the concurrent presence of global geometric (environmental) and local kinematic constraints. For this reason most available approaches decouple the treatment of these in two separate phases of the planning process. Basically we can distinguish between global and local path planning methods, which account for the two subproblems. In some approaches the configuration space of the vehicle is sampled and a graph-based description is used to grasp the topological structure of the environment [6]. For example, the Probabilistic Roadmap Method [7] uses general graphs with an offline sampling; the Rapidly Exploring Random Tree approach [8–10] builds one or more trees online to cover the free space. These sampling-based frameworks make use of a vehicle-specific local planner module (also denoted as a *steering method*), which accounts for connecting two arbitrary configurations of the nonholonomic car. Graph vertices represent configurations, while edges in the graphs correspond to local paths in these approaches. The global admissibility of these is checked by an appropriate collision detector module.

Other approaches are based on the Holonomic Path Approximation Algorithm [11], which recommends planning a preliminary holonomic (kinematically unconstrained) path first and approximating it using a steering method in a second step. In this framework, the holonomic global path is iteratively subdivided and the steering method together with a collision detector is applied to replace the parts. To ensure the convergence of this algorithm, the steering method has to verify the *topologically property*, which is defined in [12].

Steering nonholonomic systems is a difficult task even in the absence of obstacles. Exact algorithms are not available in general but only for special classes, for example, nilpotentizable [13], chain-formed [14–16], and differentially flat systems [17]. Common examples for the above-mentioned system classes include wheeled vehicles with or without trailers. There exist elegant solutions of path planning and control especially for differentially driven and car-like robots [18–21] and mobile manipulators [22–24]. Most of the available work investigates differentially driven robots with one trailer [19] or more trailers [18, 20]. The steering methods presented in [19, 20] are shown to verify the topological property as well. Since the kinematics of a car is equivalent to a differentially driven robot towing a trailer hitched on top of the wheel axis, these flatness-based methods were shown to be applicable for cars [21] as well. In these approaches a sufficiently smooth planar curve is planned for the flat output of the system, which is the axle midpoint of the last trailer, or simply the rear axle midpoint in case of a single car. Although the method proposed in [21] is a simple and elegant way of generating local paths for cars with continuous curvature, it does not take into account the upper-bounded curvature constraint (except at the start and end configurations).

An important and widely used family of steering approaches is based on optimal control. For general systems only approximate methods exist [25]. However, exact methods for computing optimal (e.g., shortest length) local paths are available for cars moving only forward (Dubins-car [26]) or both forward and backward (Reeds–Shepp-car [27–30]). For the Reeds–Shepp-car, it is shown in [27] that for any pair of configurations the shortest path can be chosen from 48 possible sets of paths, each of which consisting of maximum five circular or straight segments and having maximum two cusps. The number of sets was reduced to 46 in [28] and to 26 in [30]. Because these optimal paths consist of linear and circular segments, the resulting curvature profile contains abrupt changes, which can be traversed precisely only when the car is stopped at points where the curvature changes, in order to reorient its wheels. To overcome this limitation, a steering method is introduced in [31] which approximates Reeds–Shepp paths using continuous curvature turns, which contain clothoid curves to connect linear and circular parts. It was shown that although these paths are not optimal, they converge to the optimal Reeds–Shepp paths when the allowed sharpness of the clothoids tends to infinity. Both the Reeds–Shepp planner and its continuous curvature generalization (let us denote it as CCRS planner in the sequel) is proved to be topological admissible.

Other continuous curvature planning approaches fall into the "path smoothing" class, where a sequence of configurations or a preliminary polygonal route is given and the goal is turning it to a smooth curve. These make use of clothoids [32] and higher order polynomial curves [33–35], or curves with closed form expressions such as Bézier curves [36, 37]

or B-splines [38]. Unfortunately these are not treating issues such as topological admissibility or convergence guarantees in general planning tasks.

Beyond these approaches, there are specialized algorithms, which were designed to solve a specific planning task, for example, parallel or garage parking. These are not suited to solve more general planning problems in case of arbitrary obstacle distributions. For example, [39, 40] use circle segments and straight lines to generate trajectories for car parking. In [40] these are denoted as translation (TM) and rotation movements (RM), and a number of TM–RM–TM combinations are linked together, depending on the maneuvering complexity of the parking task. An exhaustive search is performed in every step; thus the computational complexity increases heavily with the number of necessary maneuvers. The method of Müller et al. [41] proposes solving the parking task by the general two-step path approximation approach, where they use the CCRS local planner [31] and the global planner described in [42]. Although a general framework is applied, they exploit some simplifications and assumptions (e.g., regarding the shape of obstacles and symmetry of their distribution) in order to make computations tractable for real-time applications.

A recent publication [43] introduces a planning approach which generalizes a parking planner to handle more difficult obstacle distributions, for example, navigation in a narrow parking lot containing some additional obstacles. It is shown that the algorithm can be applied in real time thanks to the fast optimization strategy it is based on. However, completeness or guaranteed convergence is not proven.

3. The RTR + TTS-Planner

Many of the mentioned planning methods were seminal for the development of our approach presented in this paper. Our goal was to design an algorithm with the following properties:

(i) It should solve the general planning task; that is, no assumptions are made regarding the obstacle distribution or the starting and goal configurations of the vehicle. At the same time, it should perform well also in the specialized parking situations.

(ii) The resulting paths should have continuous curvature profile.

(iii) The aim is to generate "human-like" paths, which are simple and contain a "meaningful" number of maneuvers needed by the task at hand.

We chose the following methodology during the design of this planning method. A two-step approximation approach similar to [11, 41] was chosen; however, our global planner already takes some nonholonomic properties of the vehicle into account. The main contributions of this paper are the introduction of our novel topological admissible continuous curvature local planning method (TTS-planner) which facilitates the construction of "human-like" paths, as well as the detailed description of the global RTR-planner.

This paper extends our previous work [45] giving a thorough theoretical treatment, detailed derivation, and proofs regarding the local planning method, as well as a more detailed description of the global planner.

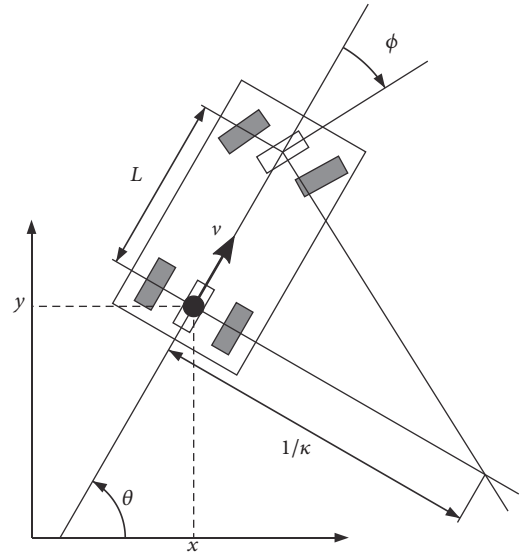


Figure 1: Kinematic model of a car.

Previous versions of the proposed two-step planning approach can be found in [46–49], where the local planner used only straight and circular segments, resulting in a discontinuous curvature profile. The currently proposed generalized version uses clothoids as well (similarly to [31]). In the derivation we rely on the results of Wilde [50] regarding the parametrization of clothoids. Our approach is mostly comparable to the CCRS planner [31] or the two-step planning method [41] based on it, because they have similar theoretical properties. However, the simulation examples in Section 6 show that our local planner generates better quality paths and has a higher success rate in narrow environments than a CCRS-based approach.

In the sequel we introduce the local TTS-planner first, by deriving it from the kinematic properties of the car and choosing path primitives with clothoids as building blocks. The global RTR-planner is introduced after that where we explain the reason of using only straight movement and in-place turning primitives in the preliminary planning phase.

4. Local Planning: The TTS-Planner

Before going into details of the local path planning method, we give mathematical definitions of the solvable problem. First the vehicle model is introduced, followed by basic properties of the feasible paths, which will serve as a basis for choosing the geometric building elements of the local planner module.

4.1. Vehicle Model. The geometrical model of a car can be seen in Figure 1. In this approach we use the one-track (or bicycle) motion model of a continuous steering kinematic car [41]:

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

where x and y are the position coordinates of the back axle midpoint (treated as the reference point of the model), θ is the orientation of the car, ϕ denotes the steering angle, and L is the distance of the front and rear wheel axles. The signed velocity of the reference point is v (positive means forward motion), and Ω stands for the steering rate. The absolute values of the steering angle and its rate of change are limited to

$$\begin{aligned} |\phi| &\le \phi_{\max} < \frac{\pi}{2} \\ |\Omega| &\le \Omega_{\max} < \infty. \end{aligned} \tag{2}$$

The instantaneous turning radius (the reciprocal of the path curvature κ) is given by

$$\frac{1}{\kappa} = \frac{L}{\tan\phi}. \tag{3}$$

The steering angle limit implies an upper bound on the path curvature

$$|\kappa| \le \kappa_{\max} = \frac{\tan\phi_{\max}}{L}. \tag{4}$$

The more widespread form of the motion equation is the following [51]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \sigma, \tag{5}$$

where σ denotes the curvature change rate. The vector $q = (x, y, \theta, \kappa)$ is called the configuration (or state) of the vehicle and $u = (v, \sigma)$ is the input vector of the system. The connection between (1) and (5) is given by

$$\begin{aligned} \kappa &= \frac{\tan\phi}{L} \\ \sigma = \dot{\kappa} &= \frac{\dot{\phi}}{L\cos^2\phi} = \frac{\Omega}{L\cos^2\phi}. \end{aligned} \tag{6}$$

An upper bound on the curvature change rate can be expressed as

$$|\sigma| \le \sigma_{\max} = \frac{\Omega_{\max}}{L} \le \frac{\Omega_{\max}}{L\cos^2\phi}. \tag{7}$$

4.2. Feasible Paths. To elaborate the properties of feasible local paths of our car model, let the translational velocity of the car be written as $v = \gamma|v|$, where $|v|$ stands for the magnitude of the velocity and γ represents the local direction of motion:

$$\gamma = \begin{cases} 1, & \text{if } v \ge 0 \text{ (forward motion)}, \\ -1, & \text{if } v < 0 \text{ (backward motion)}. \end{cases} \tag{8}$$

Furthermore, let us introduce the translational displacement of the car along its path

$$s(t) = \int_0^t |v(\tau)|\, d\tau \tag{9}$$

and express the translational velocity as

$$v = \gamma \frac{ds}{dt}. \tag{10}$$

Using (10) and the chain rule of differentiation, we reformulate the motion equation (5) as follows:

$$\frac{d}{ds}\begin{bmatrix} x \\ y \\ \theta \\ \kappa \end{bmatrix} \cdot \frac{ds}{dt} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{bmatrix} \gamma \frac{ds}{dt} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \sigma. \tag{11}$$

After dividing by (10) and defining

$$\alpha(s) = \frac{\sigma}{v} = \gamma \frac{d\kappa}{ds}(s) = \frac{d^2\theta}{ds^2}(s), \tag{12}$$

the *sharpness* of the path, we get the spatial description of feasible paths:

$$\gamma \frac{d}{ds}\begin{bmatrix} x \\ y \\ \theta \\ \kappa \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \alpha. \tag{13}$$

We can establish the following properties of these paths for our car model:

(i) The orientation θ at s is constrained by the local displacements along the x- and y-axes and by the motion direction γ. The path curvature κ is given by the first (spatial) derivative of θ and the motion direction. Thus every feasible path can be fully described by a planar curve labeled by the motion direction at every point. This fact actually arises from the differential flatness property of the car with the rear axle midpoint position as flat output [17, 21].

(ii) The sharpness of the path is the second derivative of θ (the spatial change rate of the curvature along the path) and determines the maximal allowed velocity at every point on the path:

$$|v(s)| \le \frac{\sigma_{\max}}{|\alpha(s)|}. \tag{14}$$

If a nonzero minimal traveling speed $v_{\min}$ is required, then the sharpness has an upper bound:

$$|\alpha(s)| \leq \alpha_{\max} = \frac{\sigma_{\max}}{|v_{\min}|}. \tag{15}$$

In other words, if we would like to obtain a path which can be followed by the continuous steering car (1) and we allow the car to stop only if the motion direction is reversed, then the path should be a planar curve with

(i) bounded curvature and

(ii) bounded sharpness (continuous curvature).

In Sections 4.3, 4.4, and 4.5 we are about to present specific paths fulfilling these requirements and the local path planner for generating such paths.

4.3. Path Elements. Our aim is to choose geometric primitives which can serve as building elements in the process of path construction. We assume that the motion direction is the same along a primitive, but it can change between the primitives (which means a cusp in the path). The most simple planar curves with bounded curvature are straight lines ($\kappa = 0$, called S segments in the sequel) and curvature-limited circular arcs ($|\kappa(s)| = \text{const} \leq \kappa_{\max}$, called C segments). Paths constructed by these primitives have piecewise constant curvature with abrupt curvature changes (i.e., infinite sharpness) between the pieces. To obtain continuous curvature, we are about to use *clothoids*, which are the most simple parametric curves with changing curvature and bounded sharpness.

4.3.1. Clothoids. A clothoid is a curve with constant sharpness

$$|\alpha(s)| = |\alpha| \leq \alpha_{\max} < \infty, \quad \forall s. \tag{16}$$

According to (13), the equations describing a clothoid path can be obtained by integration. If we assume that $q(0) = 0$, we get

$$x(s) = \gamma \sqrt{\frac{\pi}{|\alpha|}} C_F\left(\sqrt{\frac{|\alpha|}{\pi}} s\right), \tag{17a}$$

$$y(s) = \gamma \operatorname{sgn}(\alpha) \sqrt{\frac{\pi}{|\alpha|}} S_F\left(\sqrt{\frac{|\alpha|}{\pi}} s\right), \tag{17b}$$

where $\operatorname{sgn}(\alpha)$ denotes the sign of α and C_F and S_F stand for the Fresnel cosine and sine integrals:

$$C_F(r) = \int_0^r \cos\left(\frac{\pi}{2}\mu^2\right) d\mu, \tag{18a}$$

$$S_F(r) = \int_0^r \sin\left(\frac{\pi}{2}\mu^2\right) d\mu. \tag{18b}$$

4.3.2. Clothoid Types. From the point of view of path planning, eight different types of clothoids can be distinguished, based on the following properties:

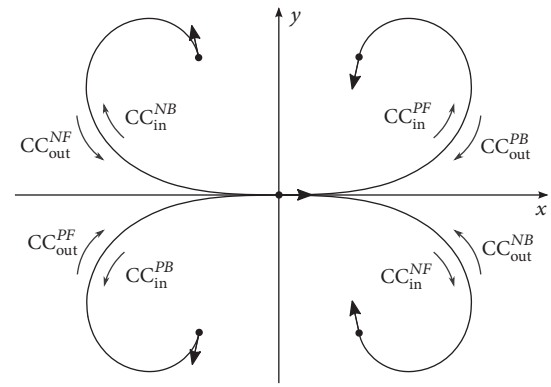


Figure 2: Clothoid types.

(i) Sign of the sharpness: $\operatorname{sgn}(\alpha) = \gamma \operatorname{sgn}(d\kappa/ds)$, it can be positive ($P, \operatorname{sgn}(\alpha) = 1$) or negative ($N, \operatorname{sgn}(\alpha) = -1$). Zero sharpness is omitted because it results in a straight or circular segment instead a clothoid.

(ii) Direction of motion: γ, it can be forward (F) or backward (B)

(iii) Sign of the absolute curvature change: $\operatorname{sgn}(d|\kappa|/ds)$. The absolute curvature grows if the vehicle is entering a turn (called *in*-type clothoid) and vanishes if it returns to a straight segment (*out*-type clothoid).

Equations (17a) and (17b) describe in-type clothoids parametrized by the constants α, γ and the running variable $s \in [0, s_{CC}]$, starting from $q_{in}(0) = 0$ and arriving at some $q_{in}(s_{CC})$, where s_{CC} denotes the whole curve length. From every in-type clothoid we can derive an out-type counterpart by preserving the same geometry but reversing the motion direction:

$$x_{out}(s) = -\gamma \sqrt{\frac{\pi}{|\alpha|}} C_F\left(\sqrt{\frac{|\alpha|}{\pi}} (s_{CC} - s)\right), \tag{19a}$$

$$y_{out}(s) = -\gamma \operatorname{sgn}(\alpha) \sqrt{\frac{\pi}{|\alpha|}} S_F\left(\sqrt{\frac{|\alpha|}{\pi}} (s_{CC} - s)\right). \tag{19b}$$

Note that these out-type clothoids start from some $q_{out}(0) = q_{in}(s_{CC})$ and arrive at $q_{out}(s_{CC}) = 0$. The eight different types of clothoids are illustrated in Figure 2.

Note that we deal here only with such clothoid path segments which have either their starting or their final configuration at the origin with zero orientation and curvature ($q(0) = 0$ for in-type and $q(s_{CC}) = 0$ for out-type clothoids). However, this causes no serious limitations. On the one hand, any $q = (x, y, \theta, 0)$ can be transformed to $q' = (0, 0, 0, 0)$ by translation and rotation. On the other hand, $\kappa(0) = 0$ or $\kappa(s_{CC}) = 0$ means that our final concatenated paths can consist of S segments, CC_{in}-C-CC_{out} triplets, and CC_{in}-CC_{out} pairs. Practically, the only limitation is that transitions between two C segments of different curvatures are allowed only with a curvature profile crossing or touching zero, that is, by a C-CC_{out}-CC_{in}-C sequence. As we will see later, this does not prohibit the creation of a complete path planner.

From now on, according to the nomenclature of [31], we will denote CC_{in}-C-CC_{out} triplets as *CC-turns* or T segments, and CC_{in}-CC_{out} pairs as *elementary paths* or E segments. Note that an elementary path is a special case of a CC-turn.

4.3.3. Reparametrization of Clothoids. By looking at (17a) and (17b), we can state that any point on an in-type clothoid starting from $q(0) = 0$ can be described by the pair $(\alpha, \gamma s)$, the sharpness and the signed displacement along the curve. It was shown by Wilde [50] that an equivalent representation is available using the pair (κ, δ), the curvature and the deflection of the curve at a certain point:

$$x(\kappa, \delta) = \operatorname{sgn}(\delta) \frac{\sqrt{2\pi |\delta|}}{\kappa} C_F\left(\sqrt{\frac{2|\delta|}{\pi}}\right), \tag{20a}$$

$$y(\kappa, \delta) = \frac{\sqrt{2\pi |\delta|}}{\kappa} S_F\left(\sqrt{\frac{2|\delta|}{\pi}}\right). \tag{20b}$$

As it can be seen, κ appears only as a multiplier in the expressions; thus it can be interpreted as a scaling factor. In this representation δ is responsible for the "shape" of the clothoid segment, and κ determines its "size."

From a computational perspective, this (κ, δ) parametrization is less convenient than the parametrization by $(\alpha, \gamma s)$. The reason is that α and γ are constant values and only s is variable along the path segment; thus the curve is described by some fixed parameters and exactly one free parameter, as usual. This makes the explicit computation of these curves simple, for example, for visualization or collision checking. Opposed to this, both κ and δ are varying along the curve, and coherency has to be maintained between them to obtain proper results. The coherency is ensured by the constant sharpness, which is related to κ and δ as

$$\alpha = \frac{\kappa^2}{2\delta}. \tag{21}$$

For simplicity, let us choose fixed curvature and deflection values κ_{CC} and δ_{CC} at the endpoint of curve. This is the point with maximal absolute curvature and deflection:

$$\kappa_{CC} = \arg\max_{\kappa} |\kappa|, \tag{22a}$$

$$\delta_{CC} = \arg\max_{\delta} |\delta|. \tag{22b}$$

Based on (20a), (20b), (22a), and (22b), in-type clothoids can be fully described by

$$x_{in}(\delta) = \operatorname{sgn}(\delta_{CC}) \frac{\sqrt{2\pi |\delta_{CC}|}}{\kappa_{CC}} C_F\left(\sqrt{\frac{2|\delta|}{\pi}}\right), \tag{23a}$$

$$y_{in}(\delta) = \frac{\sqrt{2\pi |\delta_{CC}|}}{\kappa_{CC}} S_F\left(\sqrt{\frac{2|\delta|}{\pi}}\right), \tag{23b}$$

$$\theta_{in}(\delta) = \delta, \tag{23c}$$

$$\kappa_{in}(\delta) = \kappa_{CC}\sqrt{\frac{\delta}{\delta_{CC}}}, \tag{23d}$$

where $\delta \in [0, \delta_{CC}]$ is the variable parameter and δ_{CC} and κ_{CC} are constant values. A shorter form for $x_{in}(\delta)$ and $y_{in}(\delta)$ can be obtained by defining the following functions:

$$X(\beta) = \operatorname{sgn}(\beta)\sqrt{\pi |\beta|} \cdot C_F\left(\sqrt{\frac{|\beta|}{\pi}}\right) \tag{24}$$

$$Y(\beta) = \sqrt{\pi |\beta|} \cdot S_F\left(\sqrt{\frac{|\beta|}{\pi}}\right). \tag{25}$$

Using these, (23a), (23b), (23c), and (23d) will look like

$$x_{in}(\delta) = \frac{X(2\delta)}{\kappa_{CC}}\sqrt{\frac{\delta_{CC}}{\delta}} = \frac{X(2\delta)}{\kappa_{in}(\delta)}, \tag{26a}$$

$$y_{in}(\delta) = \frac{Y(2\delta)}{\kappa_{CC}}\sqrt{\frac{\delta_{CC}}{\delta}} = \frac{Y(2\delta)}{\kappa_{in}(\delta)}, \tag{26b}$$

$$\theta_{in}(\delta) = \delta, \tag{26c}$$

$$\kappa_{in}(\delta) = \kappa_{CC}\sqrt{\frac{\delta}{\delta_{CC}}}. \tag{26d}$$

Note that if we look at the endpoint, $X(2\delta_{CC})$ and $Y(2\delta_{CC})$ can be treated as the end position of and "unscaled" in-type clothoid of δ_{CC} full deflection. The "scaling factor" $1/\kappa_{CC}$ can be used to obtain the real endpoint coordinates $x_{in}(\delta_{CC})$ and $y_{in}(\delta_{CC})$.

Now, let us consider a pair of in-type and out-type clothoids having the same geometry (such pairs were shown in Figure 2, e.g., CC_{in}^{PF} and CC_{out}^{PB}). In order to ensure the above-mentioned coherency between κ_{CC} and δ_{CC}, we define them according to (22a) and (22b) for out-type clothoids as well. However, in this case κ_{CC} belongs to the starting point, while δ_{CC} to the endpoint of the curve. Since an out-type clothoid has the same geometry as its in-type pair but opposite motion direction, we can obtain its equation as

$$x_{out}(\delta) = -\operatorname{sgn}(\delta_{CC}) \frac{\sqrt{2\pi |\delta_{CC}|}}{\kappa_{CC}} C_F\left(\sqrt{\frac{2|\delta - \delta_{CC}|}{\pi}}\right), \tag{27a}$$

$$y_{out}(\delta) = \frac{\sqrt{2\pi |\delta_{CC}|}}{\kappa_{CC}} S_F\left(\sqrt{\frac{2|\delta - \delta_{CC}|}{\pi}}\right), \tag{27b}$$

$$\theta_{out}(\delta) = \delta - \delta_{CC}, \tag{27c}$$

$$\kappa_{out}(\delta) = \kappa_{CC}\sqrt{\frac{\delta_{CC} - \delta}{\delta_{CC}}}. \tag{27d}$$

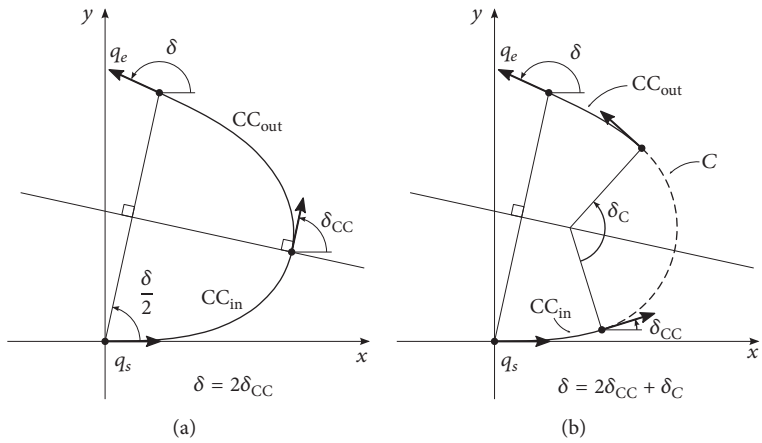


Figure 3: Structure of (a) elementary paths and (b) CC-turns.

4.3.4. Elementary Paths and CC-Turns. As already mentioned, elementary paths (E segments) are symmetric curves, consisting of a pair of compatible CC_{in} and CC_{out} clothoids. The curvature of the first curve changes from 0 to κ_{CC}, while the second from κ_{CC} back to 0. Both curves have the same deflection δ_{CC}. Compatibility is assured by the same motion direction γ and the same curvature κ_{CC} at the end of the in-type part and at the beginning of the out-type part. The compatible pairs are

(i) CC_{in}^{PF} and CC_{out}^{NF},

(ii) CC_{in}^{NF} and CC_{out}^{PF},

(iii) CC_{in}^{PB} and CC_{out}^{NB},

(iv) CC_{in}^{NB} and CC_{out}^{PB}.

A CC-turn consists of a pair of compatible CC_{in} and CC_{out} curves, with an additional circular arc between them, which has $\kappa_C = \kappa_{CC}$ curvature and δ_C deflection. As mentioned before, an elementary path is a special case of a CC-turn with $\delta_C = 0$ (see Figure 3).

Let us characterize these path components. They can be described by a parametric configuration curve $q(\delta) = (x(\delta), y(\delta), \theta(\delta), \kappa(\delta))$ with $\delta \in [0, \delta_{end}]$. The curve is fully determined by the following parameters:

(i) A starting configuration $q(0) = q_s = (x_s, y_s, \theta_s, 0)$.

(ii) The maximal curvature κ_{CC} (in an absolute sense: $|\kappa_{CC}| \geq |\kappa(\delta)|, \ \forall\delta$).

(iii) Deflections of the clothoid (δ_{CC}) and circular (δ_C) parts.

We use the following building elements to synthesize elementary paths and CC-turns:

(i) An in-type clothoid $q_{in}(\delta)$ of deflection δ_{CC}, starting from $q_{in}(0) = (0, 0, 0, 0)$ and arriving at $q_{in}(\delta_{CC}) = (x_{in}(\delta_{CC}), y_{in}(\delta_{CC}), \delta_{CC}, \kappa_{CC})$, as described by (23a), (23b), (23c), and (23d).

(ii) A circular arc $q_c(\delta)$ of deflection δ_C and curvature κ_{CC}, starting from $q_c(0) = (0, 0, 0, 0)$, described by

$$x_c(\delta) = \frac{\sin\delta}{\kappa_{CC}} \tag{28a}$$

$$y_c(\delta) = \frac{1 - \cos\delta}{\kappa_{CC}} \tag{28b}$$

$$\theta_c(\delta) = \delta \tag{28c}$$

$$\kappa_c(\delta) = \kappa_{CC}. \tag{28d}$$

(iii) A compatible out-type clothoid $q_{out}(\delta)$ of deflection δ_{CC}, starting from $q_{out}(0) = (x_{out}(0), y_{out}(0), -\delta_{CC}, \kappa_{CC})$ and arriving at $q_{out}(\delta_{CC}) = (0, 0, 0, 0)$, as described by (27a), (27b), (27c), and (27d).

The starting and end configurations of these parts have to be aligned in order to obtain the resulting elementary path or CC-turn. The alignment is performed by rotation and translation, described by homogeneous transformation matrices. Let us assign a transformation matrix to any configuration q_0 as follows:

$$q_0 = \begin{pmatrix} x_0 \\ y_0 \\ \theta_0 \\ \kappa_0 \end{pmatrix} \longrightarrow$$

$$T(q_0) = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & 0 & 0 & x_0 \\ \sin\theta_0 & \cos\theta_0 & 0 & 0 & y_0 \\ 0 & 0 & 1 & 0 & \theta_0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{29}$$

This transformation matrix represents the pose of the local coordinate frame attached to q_0. The transformation of

another configuration q by $T(q_0)$ can be performed as usual:

$$\begin{pmatrix} q' \\ 1 \end{pmatrix} = T(q_0) \begin{pmatrix} q \\ 1 \end{pmatrix}. \tag{30}$$

This means that we interpret the coordinates of q as local, relative to q_0. Multiplying by $T(q_0)$ we get q', which represents the global coordinates of the transformed q. Notice that the transformation does not affect the curvature part of the configuration. It follows that the original configuration q_0 can be obtained by applying the transformation $T(q_0)$ to $q = (0, 0, 0, \kappa_0)$:

$$\begin{pmatrix} q_0 \\ 1 \end{pmatrix} = T(q_0) \begin{pmatrix} 0 \\ 0 \\ 0 \\ \kappa_0 \\ 1 \end{pmatrix}. \tag{31}$$

The inverse transformation is given by

$$T^{-1}(q_0) = \begin{bmatrix} \cos\theta_0 & \sin\theta_0 & 0 & 0 & -x_0\cos\theta_0 - y_0\sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 & 0 & 0 & x_0\sin\theta_0 - y_0\cos\theta_0 \\ 0 & 0 & 1 & 0 & -\theta_0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{32}$$

These transformation matrices have the following chaining property:

$$^{H}q' = T(q_0) \cdot {}^{H}q \Longleftrightarrow T(q') = T(q_0) \cdot T(q), \tag{33}$$

where the upper "H" index stands for the homogenous coordinate representation $^{H}q = [q, 1]^{T}$.

Now, we have all the tools to assemble CC-turns and elementary paths. A general description of these is obtained as follows:

$$^{H}q(\delta) = T(q_s) \cdot {}^{H}q_{\text{in}}(\delta), \quad \text{if } \delta \in [0, \delta_{\text{CC}}], \tag{34a}$$

$$^{H}q(\delta) = T(q(\delta_{\text{CC}})) \cdot {}^{H}q_c(\delta - \delta_{\text{CC}}), \quad \text{if } \delta \in (\delta_{\text{CC}}, \delta_{\text{CC}} + \delta_{\text{C}}]. \tag{34b}$$

$$^{H}q(\delta) = T(q(\delta_{\text{CC}} + \delta_{\text{C}})) \cdot T^{-1}(q_{\text{out}}(0)) \cdot {}^{H}q_{\text{out}}(\delta - \delta_{\text{CC}} - \delta_{\text{C}}), \quad \text{if } \delta \in (\delta_{\text{CC}} + \delta_{\text{C}}, 2\delta_{\text{CC}} + \delta_{\text{C}}], \tag{34c}$$

where q_s stands for a given arbitrary starting configuration with zero curvature and $q_{\text{in}}(\delta)$ and $q_{\text{out}}(\delta)$ are given by (23a), (23b), (23c), (23d), (27a), (27b), (27c), and (27d), respectively. This can be treated as a "direct" approach for generating a CC-turn or elementary path. One specifies the starting configuration q_s, the maximal curvature κ_{CC}, and the deflections δ_{CC} and δ_{C}, and the end configuration $q_e = q(2\delta_{\text{CC}}+\delta_{\text{C}})$ is obtained directly using (34a), (34b), and (34c). Without loss of generality we can assume that $q_s = (0, 0, 0, 0)$. In this case we get

$$^{H}q(\delta_{\text{CC}}) = {}^{H}q_{\text{in}}(\delta_{\text{CC}}) = T(q_{\text{in}}(\delta_{\text{CC}})) \cdot {}^{H}0, \tag{35a}$$

$$^{H}q(\delta_{\text{CC}} + \delta_{\text{C}}) = T(q_{\text{in}}(\delta_{\text{CC}})) \cdot {}^{H}q_{\text{c}}(\delta_{\text{C}}) = \underbrace{T(q_{\text{in}}(\delta_{\text{CC}})) \cdot T(q_c(\delta_{\text{C}}))}_{T(q(\delta_{\text{CC}}+\delta_{\text{C}}))} \cdot {}^{H}0, \tag{35b}$$

$$^{H}q_e = {}^{H}q(2\delta_{\text{CC}} + \delta_{\text{C}}) = \underbrace{T(q(\delta_{\text{CC}} + \delta_{\text{C}})) \cdot T^{-1}(q_{\text{out}}(0))}_{T(q_e)} \cdot {}^{H}q_{\text{out}}(\delta_{\text{CC}}) = T(q_e) \cdot {}^{H}0, \tag{35c}$$

because the endpoint of a standard out-type clothoid is the origin, according to (27a), (27b), (27c), and (27d). The components of

$$T(q_e) = T(q_{\text{in}}(\delta_{\text{CC}})) \cdot T(q_c(\delta_{\text{C}})) \cdot T^{-1}(q_{\text{out}}(0)) \tag{36}$$

have the following form:

$$T(q_{\text{in}}(\delta_{\text{CC}})) = \begin{bmatrix} \cos\delta_{\text{CC}} & -\sin\delta_{\text{CC}} & 0 & 0 & \dfrac{X(2\delta_{\text{CC}})}{\kappa_{\text{CC}}} \\ \sin\delta_{\text{CC}} & \cos\delta_{\text{CC}} & 0 & 0 & \dfrac{Y(2\delta_{\text{CC}})}{\kappa_{\text{CC}}} \\ 0 & 0 & 1 & 0 & \delta_{\text{CC}} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

$$T(q_c(\delta_{\text{C}})) = \begin{bmatrix} \cos\delta_{\text{C}} & -\sin\delta_{\text{C}} & 0 & 0 & \dfrac{\sin\delta_{\text{C}}}{\kappa_{\text{CC}}} \\ \sin\delta_{\text{C}} & \cos\delta_{\text{C}} & 0 & 0 & \dfrac{1-\cos\delta_{\text{C}}}{\kappa_{\text{CC}}} \\ 0 & 0 & 1 & 0 & \delta_{\text{C}} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

In order to obtain $T^{-1}(q_{\text{out}}(0))$, we express $q_{\text{out}}(0)$ first:

$$q_{\text{out}}(0) = \begin{pmatrix} -x_{\text{in}}(\delta_{\text{CC}}) \\ y_{\text{in}}(\delta_{\text{CC}}) \\ -\delta_{\text{CC}} \\ \kappa_{\text{CC}} \end{pmatrix} = \begin{pmatrix} -\dfrac{X(2\delta_{\text{CC}})}{\kappa_{\text{CC}}} \\ \dfrac{Y(2\delta_{\text{CC}})}{\kappa_{\text{CC}}} \\ -\delta_{\text{CC}} \\ \kappa_{\text{CC}} \end{pmatrix}, \tag{38}$$

and it follows from (32) that

$$T^{-1}\left(q_{\text{out}}(0)\right) = \begin{bmatrix} \cos\delta_{\text{CC}} & -\sin\delta_{\text{CC}} & 0 & 0 & t_{1,5} \\ \sin\delta_{\text{CC}} & \cos\delta_{\text{CC}} & 0 & 0 & t_{2,5} \\ 0 & 0 & 1 & 0 & \delta_{\text{CC}} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (39)$$

where

$$\begin{aligned} t_{1,5} &= \frac{X\left(2\delta_{\text{CC}}\right)}{\kappa_{\text{CC}}}\cos\delta_{\text{CC}} + \frac{Y\left(2\delta_{\text{CC}}\right)}{\kappa_{\text{CC}}}\sin\delta_{\text{CC}}, \\ t_{2,5} &= \frac{X\left(2\delta_{\text{CC}}\right)}{\kappa_{\text{CC}}}\sin\delta_{\text{CC}} - \frac{Y\left(2\delta_{\text{CC}}\right)}{\kappa_{\text{CC}}}\cos\delta_{\text{CC}}. \end{aligned} \quad (40)$$

After multiplying the three matrices we get

$$T(q_e) = \begin{bmatrix} \cos\left(\delta_{\text{end}}\right) & -\sin\left(\delta_{\text{end}}\right) & 0 & 0 & \dfrac{A\left(2\delta_{\text{CC}},\delta_C\right)}{\kappa_{\text{CC}}} \\ \sin\left(\delta_{\text{end}}\right) & \cos\left(\delta_{\text{end}}\right) & 0 & 0 & \dfrac{B\left(2\delta_{\text{CC}},\delta_C\right)}{\kappa_{\text{CC}}} \\ 0 & 0 & 1 & 0 & \delta_{\text{end}} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (41)$$

where

$$\delta_{\text{end}} = 2\delta_{\text{CC}} + \delta_C, \quad (42)$$

$$\begin{aligned} A\left(2\delta_{\text{CC}},\delta_C\right) &= X\left(2\delta_{\text{CC}}\right)\left(1+\cos\left(2\delta_{\text{CC}}+\delta_C\right)\right) \\ &\quad + Y\left(2\delta_{\text{CC}}\right)\sin\left(2\delta_{\text{CC}}+\delta_C\right) \\ &\quad + \sin\left(\delta_{\text{CC}}+\delta_C\right) - \sin\delta_{\text{CC}}, \end{aligned} \quad (43)$$

$$\begin{aligned} B\left(2\delta_{\text{CC}},\delta_C\right) &= X\left(2\delta_{\text{CC}}\right)\sin\left(2\delta_{\text{CC}}+\delta_C\right) \\ &\quad + Y\left(2\delta_{\text{CC}}\right)\left(1-\cos\left(2\delta_{\text{CC}}+\delta_C\right)\right) \\ &\quad - \cos\left(\delta_{\text{CC}}+\delta_C\right) + \cos\delta_{\text{CC}}. \end{aligned} \quad (44)$$

The end configuration of the CC-turn is given by the elements in the last column of $T(q_e)$:

$$x_e = \frac{A\left(2\delta_{\text{CC}},\delta_C\right)}{\kappa_{\text{CC}}}. \quad (45a)$$

$$y_e = \frac{B\left(2\delta_{\text{CC}},\delta_C\right)}{\kappa_{\text{CC}}} \quad (45b)$$

$$\theta_e = 2\delta_{\text{CC}} + \delta_C \quad (45c)$$

$$\kappa_e = 0 \quad (45d)$$

It is important to note that, in order to fully define a CC-turn, two of the deflections δ_{CC}, δ_C, and δ_{end} should be given, together with the maximal curvature κ_{CC}. The third deflection parameter is determined by (42).

The equations of an elementary path are similar to (45a), (45b), (45c), and (45d); we only need to substitute $\delta_C = 0$:

$$x_e = \frac{A\left(2\delta_{\text{CC}},0\right)}{\kappa_{\text{CC}}} = \frac{A\left(2\delta_{\text{CC}}\right)}{\kappa_{\text{CC}}}, \quad (46a)$$

$$y_e = \frac{B\left(2\delta_{\text{CC}},0\right)}{\kappa_{\text{CC}}} = \frac{B\left(2\delta_{\text{CC}}\right)}{\kappa_{\text{CC}}}, \quad (46b)$$

$$\theta_e = 2\delta_{\text{CC}}, \quad (46c)$$

$$\kappa_e = 0, \quad (46d)$$

where

$$\begin{aligned} A\left(2\delta_{\text{CC}}\right) &= X\left(2\delta_{\text{CC}}\right)\left(1+\cos\left(2\delta_{\text{CC}}\right)\right) \\ &\quad + Y\left(2\delta_{\text{CC}}\right)\sin\left(2\delta_{\text{CC}}\right), \\ B\left(2\delta_{\text{CC}}\right) &= X\left(2\delta_{\text{CC}}\right)\sin\left(2\delta_{\text{CC}}\right) \\ &\quad + Y\left(2\delta_{\text{CC}}\right)\left(1-\cos\left(2\delta_{\text{CC}}\right)\right). \end{aligned} \quad (47)$$

4.4. TTS Paths for Local Planning. Algorithms that generate feasible local paths for a wheeled system—such as our car model (5)—in the absence of obstacles are called local planners or steering methods. The exact definition of these is the following.

Definition 1 (steering method). Let us denote by Λ_f the set of all feasible local paths λ_f regarding system equation $\dot{q} = f(q,u)$. A steering method (or local planner module) is a function

$$\text{Steer} : \mathscr{C} \times \mathscr{C} \longrightarrow \Lambda_f, \quad \text{i.e.,} \lambda_f(\cdot) = \text{Steer}\left(q_0, q_1\right) \quad (48)$$

such that

$$\begin{aligned} \lambda_f(0) &= \text{Steer}\left(q_0,q_1\right)(0) = q_0, \\ \lambda_f\left(S_\lambda\right) &= \text{Steer}\left(q_0,q_1\right)\left(S_\lambda\right) = q_1. \end{aligned} \quad (49)$$

As already mentioned in Section 2, the most widely used local planners for the simple kinematic car model (without the requirement of curvature continuity) are based on the optimal length Reeds–Shepp (RS) paths [27]. These local paths consist of maximum five circular or straight segments and have at most two cusps, where the curvature of circular arcs is exactly $\pm\kappa_{\max}$. If the continuous steering car model (5) is used, the continuous curvature generalization proposed in [31] can be applied. This approximates Reeds–Shepp paths by replacing circular segments with CC-turns. It was shown that although the resulting CCRS paths are not optimal, they converge to the optimal RS paths when the allowed sharpness of the clothoids tends to infinity. The CC-turns in this approach consist of clothoids having the maximal allowed sharpness and circle segments of maximal curvature.

In the sequel we are doing something similar: introduce a new set of local paths for the continuous steering car. The main ideas which motivated the invention of our approach instead using the CCRS paths are the following:

(i) The main goal is global planning in the presence of obstacles. Most existing planning methods use a concatenation of local paths, which has the consequence that the result will be not optimal, even if the parts were generated by an optimal planner. Even the CCRS paths are suboptimal. Thus, having the final application in our mind, we lifted the requirement of optimality of our local paths.

(ii) Instead minimizing the global path length, we had the goal to achieve "natural" or "human-like" paths. What this means is informally described in the following points.

(iii) The RS (resp., CCRS) paths consist of up to five circular (resp., CC-turn) and straight segments. We show in the sequel that three path segments are sufficient to connect any two configurations.

(iv) The RS (resp., CCRS) paths use circular segments (resp., CC-turns) having the maximal allowed curvature (the steering wheel is turned to the end position in every curve). Our planner is not restricted to this. Only the limit is taken into account, but smaller curvatures are allowed.

(v) The last segment of RS (resp., CCRS) paths is always circular (resp., CC-turn). We argue that this contradicts the behavior of human drivers. Because the car can only move forward or backward, in most cases the more natural behavior is to arrive at a given goal configuration by a straight movement. There are a few exceptions, for example, the case of parallel parking, but these can usually be transformed to the mentioned case by reversing the roles of initial and goal configurations. Consequently, we choose a straight segment as the last section of our local paths.

To simplify notation in the sequel, we will denote local paths as a concatenation of S, T, and E letters, corresponding straight segments, CC-turns, and elementary paths, respectively. In our approach we use TTS (as well as EES) paths (note that in our previous work [45] we used the notations T^*TS and E^*ES where T^* and E^* represented "CC-turn or straight segment" and "elementary path or straight segment," although we omit here the possibility of having a straight segment as the first element because our experiments showed that STS and SES paths are chosen very rarely by the implemented planning algorithm). In the sequel we present the construction of EES paths first (where we use only elementary paths in place of CC-turns) and show that this class of paths is sufficient to connect an arbitrary pair of configurations. After that we generalize E segments to CC-turns.

4.4.1. Construction of EES Paths. Let us assume without loss of generality that the path starts from a given initial configuration $q_I = (x_I, y_I, \theta_I, 0)$ and arrives at the origin as goal configuration $q_G = (0, 0, 0, 0)$. Any query pair (q_I, q_G) can be transformed to this form by translation and rotation, and the designed path can be transformed back. Let us denote the intermediate configurations between the path segments

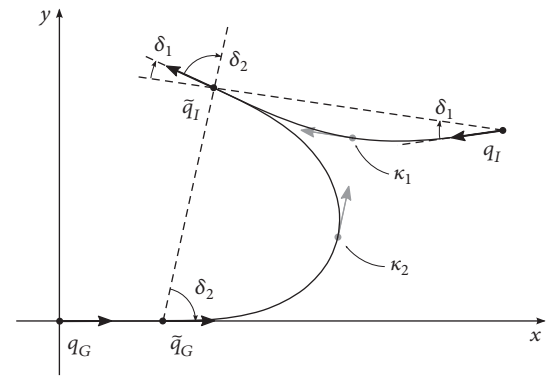


Figure 4: *EES* path.

by $\tilde{q}_I$ and $\tilde{q}_G$. For every segment, i denotes its index (1, 2 or 3) inside the path. A straight segment is parametrized by its absolute length s_i and direction γ_i and an elementary path segment by its curvature $\kappa_{i,\mathrm{CC}}$ and deflection $\delta_{i,\mathrm{CC}}$ at the middle point between the $\mathrm{CC}_{\mathrm{in}}$ and $\mathrm{CC}_{\mathrm{out}}$ parts. To simplify the notation, let us neglect the "CC" subscript from now on: let $\kappa_i = \kappa_{i,\mathrm{CC}}$ and $\delta_i = \delta_{i,\mathrm{CC}}$.

The scheme of an EES path (illustrated in Figure 4) is the following:

$$q_I \xrightarrow[\delta_1,\kappa_1]{E} \tilde{q}_I \xrightarrow[\delta_2,\kappa_2]{E} \tilde{q}_G \xrightarrow[s_3,\gamma_3]{S} q_G. \tag{50}$$

The coordinates of $\tilde{q}_I$ can be obtained from q_I by a transformation belonging to the first elementary path

$$^{H}\tilde{q}_I = T(q_I) \cdot T(q_e(\delta_1, \kappa_1)) \cdot {}^{H}0, \tag{51}$$

where

$$T(q_e(\delta_1, \kappa_1)) = \begin{bmatrix} \cos(2\delta_1) & -\sin(2\delta_1) & 0 & 0 & \frac{A(2\delta_1)}{\kappa_1} \\ \sin(2\delta_1) & \cos(2\delta_1) & 0 & 0 & \frac{B(2\delta_1)}{\kappa_1} \\ 0 & 0 & 1 & 0 & 2\delta_1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T(q_I) = \begin{bmatrix} \cos(\theta_I) & -\sin(\theta_I) & 0 & 0 & x_I \\ \sin(\theta_I) & \cos(\theta_I) & 0 & 0 & y_I \\ 0 & 0 & 1 & 0 & \theta_I \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{52}$$

By applying the matrix product we obtain

$$\tilde{x}_I = \frac{A(2\delta_1)}{\kappa_1}\cos(\theta_I) - \frac{B(2\delta_1)}{\kappa_1}\sin(\theta_I) + x_I \tag{53a}$$

$$\tilde{y}_I = \frac{A(2\delta_1)}{\kappa_1}\sin(\theta_I) + \frac{B(2\delta_1)}{\kappa_1}\cos(\theta_I) + y_I \tag{53b}$$

$$\widetilde{\theta}_I = 2\delta_1 + \theta_I \tag{53c}$$

$$\widetilde{\kappa}_I = 0. \tag{53d}$$

Let us define the auxiliary functions

$$\begin{aligned} C(\beta,\psi) &= A(\beta)\cos\psi - B(\beta)\sin\psi \\ D(\beta,\psi) &= A(\beta)\sin\psi + B(\beta)\cos\psi \end{aligned} \tag{54}$$

and write (53a), (53b), (53c), and (53d) in a shorter form:

$$\widetilde{x}_I = \frac{C(2\delta_1,\theta_I)}{\kappa_1} + x_I \tag{55a}$$

$$\widetilde{y}_I = \frac{D(2\delta_1,\theta_I)}{\kappa_1} + y_I \tag{55b}$$

$$\widetilde{\theta}_I = 2\delta_1 + \theta_I \tag{55c}$$

$$\widetilde{\kappa}_I = 0. \tag{55d}$$

Alternatively, we can get $\widetilde{q}_I$ from $\widetilde{q}_G = (\widetilde{x}_G, 0, 0, 0)$ by "traveling backward" along the elementary path. This corresponds to a normal E segment with inverse deflection. We can obtain $\widetilde{q}_I$ this way by applying the transformation

$${}^{H}\widetilde{q}_I = T(\widetilde{q}_G)\cdot T(q_e(-\delta_2,\kappa_2))\cdot {}^{H}0, \tag{56}$$

where $T(q_e(-\delta_2,\kappa_2))$ is the same as (41) with parameters $\delta_{CC} = -\delta_2$ and $\delta_C = 0$:

$$\begin{aligned} &T(q_e(-\delta_2,\kappa_2)) \\ &= \begin{bmatrix} \cos(2\delta_2) & \sin(2\delta_2) & 0 & 0 & \dfrac{A(-2\delta_2)}{\kappa_2} \\ -\sin(2\delta_2) & \cos(2\delta_2) & 0 & 0 & \dfrac{B(-2\delta_2)}{\kappa_2} \\ 0 & 0 & 1 & 0 & -2\delta_2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{57}$$

and $T(\widetilde{q}_G)$ looks like

$$T(\widetilde{q}_G) = \begin{bmatrix} 1 & 0 & 0 & 0 & \widetilde{x}_G \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{58}$$

The result of the transformation is

$$\widetilde{x}_I = \frac{A(-2\delta_2)}{\kappa_2} + \widetilde{x}_G \tag{59a}$$

$$\widetilde{y}_I = \frac{B(-2\delta_2)}{\kappa_2} \tag{59b}$$

$$\widetilde{\theta}_I = -2\delta_2 \tag{59c}$$

$$\widetilde{\kappa}_I = 0. \tag{59d}$$

By comparing (55a), (55b), (55c), (55d), (59a), (59b), (59c), and (59d) we can conclude that the path parameters δ_1, κ_1, δ_2, κ_2, and $\widetilde{x}_G = -\gamma_3 s_3$ of an *EES* path fulfill the following constraints:

$$\frac{C(2\delta_1,\theta_I)}{\kappa_1} + x_I = \frac{A(-2\delta_2)}{\kappa_2} - \gamma_3 s_3 \tag{60a}$$

$$\frac{D(2\delta_1,\theta_I)}{\kappa_1} + y_I = \frac{B(-2\delta_2)}{\kappa_2} \tag{60b}$$

$$2\delta_1 + \theta_I = -2\delta_2. \tag{60c}$$

One can conclude that in an *EES* planning problem only one path segment can be parametrized freely; the others are determined. In the sequel we will use δ_1 and κ_1 as free parameters.

Let us pay some attention to the ranges of angle parameters in the above equations. The orientation parameter θ_I is treated as being in the interval $[-\pi,\pi]/\sim$, where "$/\sim$" means that the two boundary values are identified. Similarly to the geometric derivation of CC-turns, we introduce an upper bound $|\delta_{\mathrm{end},i}| \le \pi$ on the whole deflection of both elementary paths, which results in $\delta_i \in [-\pi/2,\pi/2]$. In order to ensure that δ_1, δ_2, and θ_I being in the given intervals fulfill the constraint (60c), we have to choose the set carefully from which δ_1 and δ_2 can be taken. Let us examine two cases:

(i) $\theta_I \in [0,\pi]$: choosing $\delta_1 \in [-\pi/2, \pi/2 - \theta_I/2] \subseteq [-\pi/2,\pi/2]$ results in

$$\delta_1 + \frac{\theta_I}{2} = -\delta_2 \in \left[-\frac{\pi}{2} + \frac{\theta_I}{2}, \frac{\pi}{2}\right] \subseteq \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]. \tag{61}$$

(ii) $\theta_I \in [-\pi,0]$: choosing $\delta_1 \in [-\theta_I/2 - \pi/2, \pi/2] \subseteq [-\pi/2,\pi/2]$ results in

$$\delta_1 + \frac{\theta_I}{2} = -\delta_2 \in \left[-\frac{\pi}{2}, \frac{\pi}{2} + \frac{\theta_I}{2}\right] \subseteq \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]. \tag{62}$$

These choices ensure both δ_1 and δ_2 being in $[-\pi/2,\pi/2]$ for any values of θ_I. We can write these conditions in a single compact form:

$$\delta_1, \delta_2 \in I_\delta(\theta_I) = \left[-\mathrm{sgn}_p(\theta_I)\frac{\pi}{2}, \mathrm{sgn}_p(\theta_I)\frac{\pi}{2} - \frac{\theta_I}{2}\right], \tag{63}$$

where $\mathrm{sgn}_p(\cdot)$ is a special sign function with $\mathrm{sgn}_p(0) = 1$:

$$\mathrm{sgn}_p(x) = \begin{cases} \mathrm{sgn}(x), & \text{if } x \neq 0 \\ 1, & \text{if } x = 0 \end{cases}. \tag{64}$$

An illustration of this set can be seen in Figure 5.

Up to now we have not taken the curvature bound into account. It can be proven that an arbitrary configuration pair can be connected by an *EES* path even if an upper bound on the absolute curvature is given. The following Lemma states this fact; the proof is elaborated in Appendix A.

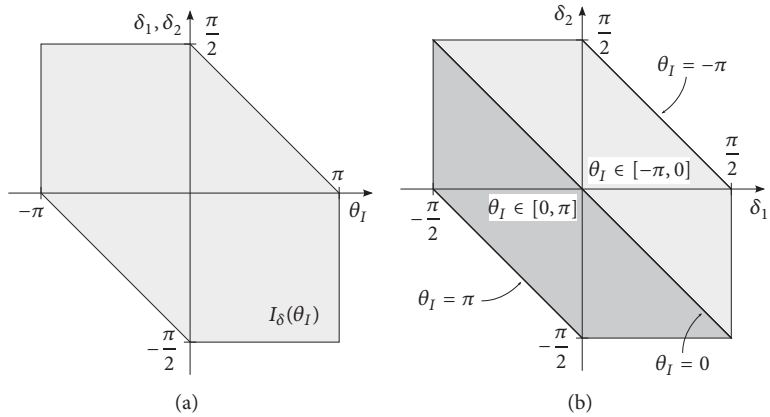


FIGURE 5: Set of valid (δ_1, δ_2) pairs for *EES* paths.

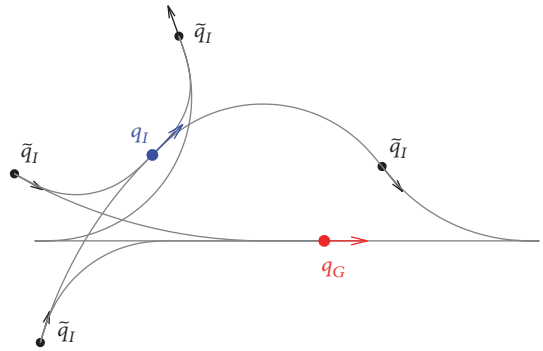


FIGURE 6: There is an infinite number of *TTS* solutions between two configurations.

Lemma 2. *In the absence of obstacles, any (q_I, q_G) pair of initial and goal configurations of a continuous steering car can be connected by an EES path.*

Furthermore, it turns out that infinitely many solutions can be found even with bounded curvature, due to the infinitely many choices of δ_1 and κ_1 parameters. This means that $\tilde{q}_I$ can be chosen in infinitely many ways, as illustrated in Figure 6.

4.4.2. From EES to TTS Paths. As we have already seen in Section 4.3.4, elementary paths are special CC-turns with $\delta_C = 0$. It follows that the results for elementary paths used in *EES* planning can be generalized to CC-turns as well. The equations of an elementary path starting from any $\tilde{q}_I$ and arriving at the x-axis—the middle segment of an *EES* path—are given by (59a), (59b), (59c), and (59d). This can easily be generalized to the case when the deflection of the circular part is nonzero:

$$\tilde{x}_I = \frac{A\left(-2\delta_{2,\mathrm{CC}}, -\delta_{2,C}\right)}{\kappa_2} + \tilde{x}_G \tag{65a}$$

$$\tilde{y}_I = \frac{B\left(-2\delta_{2,\mathrm{CC}}, -\delta_{2,C}\right)}{\kappa_2} \tag{65b}$$

$$\tilde{\theta}_I = -2\delta_{2,\mathrm{CC}} - \delta_{2,C} \tag{65c}$$

$$\tilde{\kappa}_I = 0, \tag{65d}$$

where $\delta_{2,\mathrm{CC}}$ and $\delta_{2,C}$ stand for the deflections of the clothoid and the circular parts, respectively. The full deflection of the CC-turn is

$$\delta_{2,\mathrm{end}} = 2\delta_{2,\mathrm{CC}} + \delta_{2,C} = -\tilde{\theta}_I. \tag{66}$$

Let us introduce the relative amount of the circular part of a CC-turn as

$$R_\delta = \frac{\delta_C}{\delta_{\mathrm{end}}}. \tag{67}$$

Using this, we can express (65b) as

$$\tilde{y}_I = \frac{B\left(\left(1 - R_{\delta,2}\right)\delta_{2,\mathrm{end}}, R_{\delta,2}\delta_{2,\mathrm{end}}\right)}{\kappa_2}. \tag{68}$$

The maximal curvature and the sharpness of the CC-turn are given by

$$\kappa_2 = \frac{1}{\tilde{y}_I} B\left(\left(1 - R_{\delta,2}\right)\delta_{2,\mathrm{end}}, R_{\delta,2}\delta_{2,\mathrm{end}}\right) \tag{69a}$$

$$\alpha_2 = \frac{\kappa_2^2}{2\delta_{2,\mathrm{CC}}} = \frac{B^2\left(\left(1 - R_{\delta,2}\right)\delta_{2,\mathrm{end}}, R_{\delta,2}\delta_{2,\mathrm{end}}\right)}{\tilde{y}_I^2\left(1 - R_{\delta,2}\right)\delta_{2,\mathrm{end}}}. \tag{69b}$$

Note that if we fix the full deflection $\delta_{2,\mathrm{end}}$ and change the ratio $R_{\delta,2}$ only, then for a given $\tilde{q}_I$ the end configuration $\tilde{q}_G$ on the x-axis remains the same; only κ_2 and α_2 are affected. In other words, we can tune the shape of a CC-turn with R_δ without affecting the starting and final configurations. This is illustrated in Figure 7, where clothoids are drawn with solid black lines and the red dashed arcs represent circular segments.

When R_δ grows, the curvature is decreasing but the sharpness is increasing at the same time, independently from the full deflection of the path segment. This property can be examined in Figure 8, which shows surface and contour plots of the (absolute) curvature and sharpness of a CC-turn (with

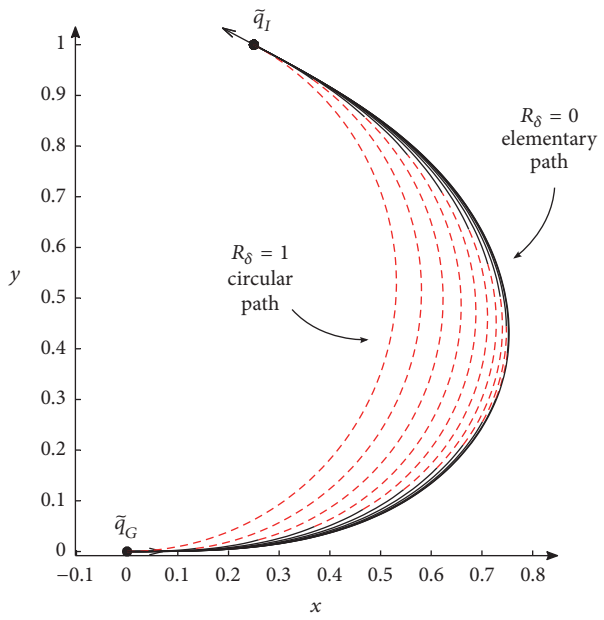


FIGURE 7: Effect of R_δ on the shape of a CC-turn (R_δ is modified in steps of 0.1).

$\tilde{y}_I = 1$), depending on the full deflection and the relative amount of the circular part.

It follows that if we already have obtained an E segment between any two configurations, we can reshape it to a CC-turn having smaller curvature and higher sharpness by tuning the R_δ parameter. With this, we can define a maximal sharpness $\alpha_{\max}$ beyond the already treated $\kappa_{\max}$ constraint. If the sharpness of the initial E segment is less than $\alpha_{\max}$, then there exists an R_δ value such that $|\alpha| = \alpha_{\max}$. This is beneficial because the length of the path segment is decreasing, as can be seen in Figure 7.

If we have an *EES* path, we can do this reshaping for every E segment to obtain a *TTS* path. However, if any of the E segments violates the $\alpha_{\max}$ constraint, then it is not so straightforward. Fortunately it can be shown that a *TTS* solution obeying the $\alpha_{\max}$ constraint exists in all cases. If the second segment in the *EES* solution violates the sharpness constraint, we can write using (59c) and (69b) and (A.3) in Appendix A:

$$\kappa'_{\max} = \sqrt{\alpha_{\max}\tilde{\theta}_I} \tag{70}$$

$$|\tilde{y}_I| \geq \frac{B\left(\tilde{\theta}_I\right)}{\kappa'_{\max}}, \tag{71}$$

where $\kappa'_{\max}$ is the minimal value of the curvature upper bound ensuring that the second elementary path will have $|\alpha| \leq \alpha_{\max}$. If we would like to obtain a *TTS* path from an *EES* solution, then by recalling (55b) and (A.6) in Appendix A we can state that κ_1 can be decreased arbitrarily in order to fulfill condition (71) for the second segment and the maximal sharpness constraint for the first segment at the same time.

The only drawback of using *TTS* paths instead *EES* solutions is that the sharpness tuning requires an iterative search of R_δ in the interval $[0, 1]$. This cannot be avoided, because R_δ is in the argument of the $B(\cdot, \cdot)$ function in (69a) and (69b), which does not have an inverse because of the contained Fresnel integrals.

4.5. A Steering Method Based on EES Paths. A *TTS* path planner based on the conditions in proof of Lemma 2 and their generalization in Section 4.4.2 is not a steering method itself (in the sense of Definition 1), because it delivers not only one solution for a query pair. Instead of this, it delivers a parametrized class of solutions, which contains an infinite number of *EES* (and *TTS*) paths from q_I to q_G, fulfilling the maximum curvature (and sharpness) constraint. This property is useful in global planning problems (i.e., in the presence of obstacles), because one can more likely find a collision-free path from a solution class than from the one and only solution of an exact local planner.

This does not mean that in the presence of obstacles the existence of any collision-free (local) solution would be guaranteed. For successful global planning explicit reasoning about free space connectivity is necessary. As mentioned in Section 3, we apply the framework of the Holonomic Path Approximation Algorithm [11] for this purpose. A distinct global planner delivers a preliminary geometric path, which is iteratively subdivided and approximated by an appropriate steering method. To ensure the convergence of this algorithm, the local planner has to generate such paths that get closer to the original path as the local endpoints get closer to each other. This is called the *topological property* [12] which is formulated more precisely in Appendix B.

Therefore, if a collision-free geometric path is given and we have an appropriate steering method that verifies the topological property, then a complete (completeness means that the algorithm does not fail to return a solution if the problem is solvable) approximation algorithm can be constructed (which was a footnote in the manuscript). Hence, in order to design such a complete algorithm based on *TTS* paths, we need a steering method which returns exactly one from the class of *TTS* paths and verifies the topological property.

4.5.1. Definition of e$\bar{e}$S Paths. In the sequel we take only *EES* paths into account for the purpose of designing the exact steering method. This choice will make possible proving the topological property, because *EES* paths have closed form expressions. In contrast to that, *TTS* paths with a given sharpness can only be obtained by iterative deriving from *EES* solutions, as seen in Section 4.4.2.

More possibilities can be found to restrict the class of *EES* paths to exactly one specific solution for every query pair (q_I, q_G). For example, we may choose the sign of κ_2 and the equality in condition (A.4) in Appendix A and fix δ_1 to any chosen value except 0 and $-\theta_I \pm \pi$. In this case the remaining parameter κ_1 is determined unequivocally. This would result in an *EES* path with $|\kappa_2| = \kappa_{\max}$ and having the intermediate orientation $\tilde{\theta}_I = \theta_I + 2\delta_1$ at the end of the first E segment. This approach has the problem that it is hard to formulate any reasonable direct rule for choosing δ_1 and the sign of κ_2 for a given query. Instead of this we

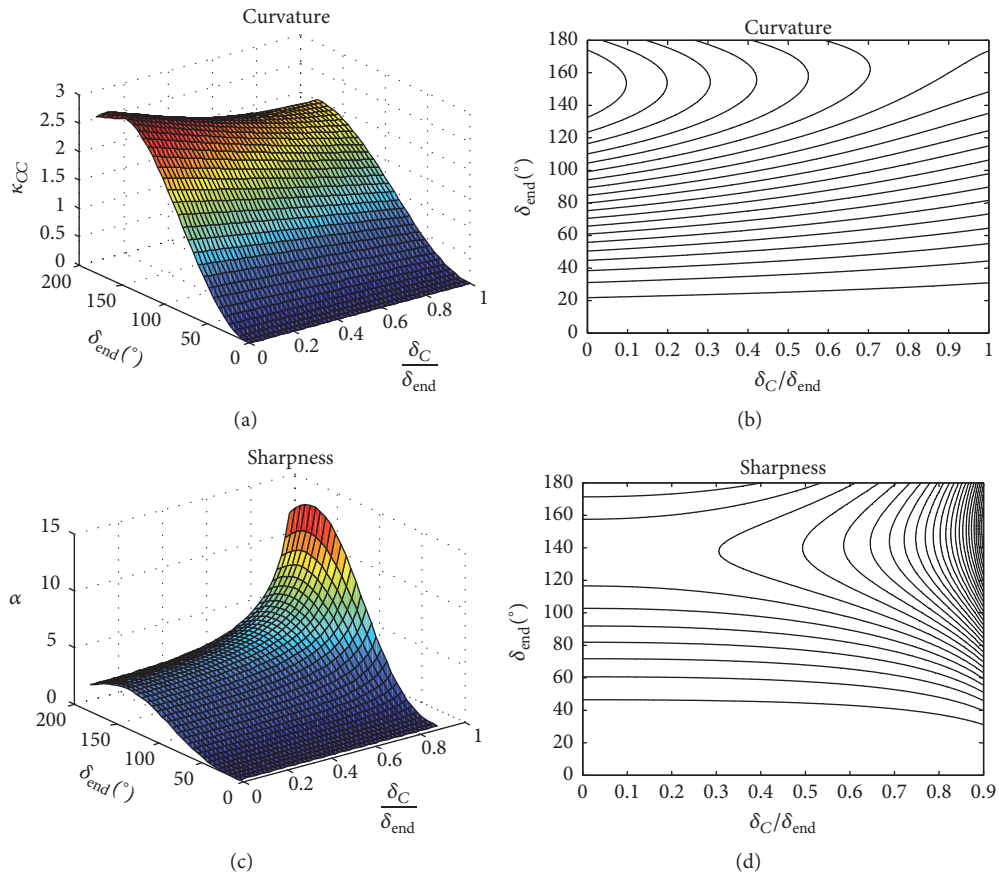


FIGURE 8: Dependence of (a)-(b) maximal curvature and (c)-(d) sharpness of CC-turns on the full detection and relative amount of the circular part (absolute values are depicted).

introduce the following approach. Let us choose $\kappa_1 = -\kappa_2$ such that $|\kappa_1| = |\kappa_2|$ is maximal. In this case we do not have to specify any other parameters. The resulting *EES* path has two E segments with same maximal absolute curvature, but having opposite signs (the steering wheel is turned from right to left or reversely). This type of paths is denoted as $e\bar{e}S$ paths in the sequel to emphasize that the resulting turning radii (reciprocal of the curvature) are opposite, equal, and minimal. The $e\bar{e}S$ planning problem is formulated as follows:

$$q_I \xrightarrow[\delta_1,\kappa_1]{E} \widetilde{q}_I \xrightarrow[\delta_2,\kappa_2]{E} \widetilde{q}_G \xrightarrow[s_3,\gamma_3]{S} q_G \tag{72}$$

such that

$$\begin{aligned} q_I &= (x_I, y_I, \theta_I, 0), \\ q_G &= (0, 0, 0, 0), \end{aligned} \tag{73}$$

$$\kappa := \kappa_1 = -\kappa_2, \tag{74}$$

$$|\kappa| \le \kappa_{\max}, \tag{75}$$

$$|\kappa| \longrightarrow \max. \tag{76}$$

Using (74) we can reformulate (60a), (60b), and (60c) as

$$x_I + \gamma_3 s_3 + \frac{A(2\delta_1 + \theta_I) + C(2\delta_1, \theta_I)}{\kappa} = 0 \tag{77a}$$

$$y_I + \frac{\overbrace{B(2\delta_1 + \theta_I) + D(2\delta_1, \theta_I)}^{G(2\delta_1,\theta_I)}}{\kappa} = 0 \tag{77b}$$

$$\delta_2 = -\delta_1 - \frac{\theta_I}{2}. \tag{77c}$$

The necessary and sufficient condition for existence of this kind of *EES* paths is obtained by (75) and (77b):

$$\left|G(2\delta_1, \theta_I)\right| = \left|y_I\right| \cdot |\kappa| \le \left|y_I\right| \cdot \kappa_{\max}. \tag{78}$$

In the sequel we use some properties of the $G(\cdot,\cdot)$ function, which are established in Appendix C.

Lemma 3. *For any* (y_I, θ_I) *there exists* δ_1 *such that* $|G(2\delta_1, \theta_I)| \le |y_I|\kappa_{\max}$.

Proof. According to Properties C.6 and C.7 of the G function we can state that for all θ_I there exist δ_1 values for both $G(2\delta_1, \theta_I) \neq 0$ and $G(2\delta_1, \theta_I) = 0$. Since G is a continuous function, this means that, for all θ_I, $|G(2\delta_1, \theta_I)|$ can take an arbitrarily small value. □

4.5.2. The $e\bar{e}S$ Steering Method. Based on the existence condition (78) and the optimization criterion (76) we can construct a steering method which generates a single $e\bar{e}S$ path for any local planning query. We denote it as the *$e\bar{e}S$-planner*.

Let us examine the special case $y_I = 0$ first, where the existence condition (78) takes the form

$$\left|G\left(2\delta_1, \theta_I\right)\right| = 0. \tag{79}$$

According to the Proof of Property C.7 in Appendix, we can state that $|G(2\delta_1, \theta_I)|$ has a zero for all θ_I in the interval $\delta_1 \in [-\theta_I/2, 0]$. Let us denote this δ_1 solution as $\delta_{1,z}$:

$$\delta_{1,z}\left(\theta_I\right) : \left\{\delta_1 \in \left[-\frac{\theta_I}{2}, 0\right] \mid G\left(2\delta_1, \theta_I\right) = 0\right\} \tag{80}$$

In this case κ could be chosen arbitrarily, but the condition (76) suggests the choice of $\kappa = \pm\kappa_{\max}$.

If $y_I \neq 0$ then it follows that

$$\kappa = -\frac{G\left(2\delta_1, \theta_I\right)}{y_I}, \tag{81}$$

and δ_1 which maximizes $|\kappa|$ is defined by

$$\delta_1^*\left(\theta_I\right) = \arg \max_{\delta_1 \in I_\delta(\theta_I)} \left\{\left|G\left(2\delta_1, \theta_I\right)\right|\right\}. \tag{82}$$

Let us define κ^* as

$$\kappa^*\left(y_I, \theta_I\right) = -\frac{G\left(2\delta_1^*\left(\theta_I\right), \theta_I\right)}{y_I}, \tag{83}$$

which can be treated as the unbounded optimum of κ. Since $|G|$ has a maximum at δ_1^* and zero at $\delta_{1,z}$, an arbitrary $|\kappa| \in (0, |\kappa^*|]$ can be achieved by $\delta_1 \in (\delta_{1,z}, \delta_1^*]$. It follows that if $|\kappa^*(y_I, \theta_I)| > \kappa_{\max}$, then a $\delta_{1,\max}(y_I, \theta_I)$ can be found between $\delta_1^*(\theta_I)$ and $\delta_{1,z}(\theta_I)$ such that $|\kappa| = \kappa_{\max}$:

$$\begin{aligned} &\delta_{1,\max}\left(y_I, \theta_I\right) : \\ &\left\{\delta_1 \in \left(\delta_{1,z}, \delta_1^*\right] \mid \frac{\left|G\left(2\delta_1, \theta_I\right)\right|}{\left|y_I\right|} = \kappa_{\max}\right\} \end{aligned} \tag{84}$$

We can summarize the $e\bar{e}S$ steering method as follows. For a given q_I we have to determine δ_1 and $\kappa = \kappa_1$ first:

Condition	δ_1	κ_1
$y_I = 0$	$\delta_{1,z}\left(\theta_I\right)$	$\pm\kappa_{\max}$
$y_I \neq 0, \quad \left\|\kappa^*\right\| \leq \kappa_{\max}$	$\delta_1^*\left(\theta_I\right)$	$\kappa^*\left(y_I, \theta_I\right)$
$y_I \neq 0, \quad \left\|\kappa^*\right\| > \kappa_{\max}$	$\delta_{1,\max}\left(y_I, \theta_I\right)$	$\operatorname{sgn}\left(\kappa^*\right)\kappa_{\max}$

(85)

The remaining path parameters δ_2, κ_2, and $\gamma_3 s_3$ can be determined using (77c), (74), and (77a), respectively.

We have proven that the above defined $e\bar{e}S$ steering method verifies the topological property which renders it suitable for application in any approximation-based planning approach. The topological property theoretically guarantees convergence of the approximation process in any cases. This advantageous fact is stated by Theorem B.2 in Appendix B and the proof is given there in detail as well.

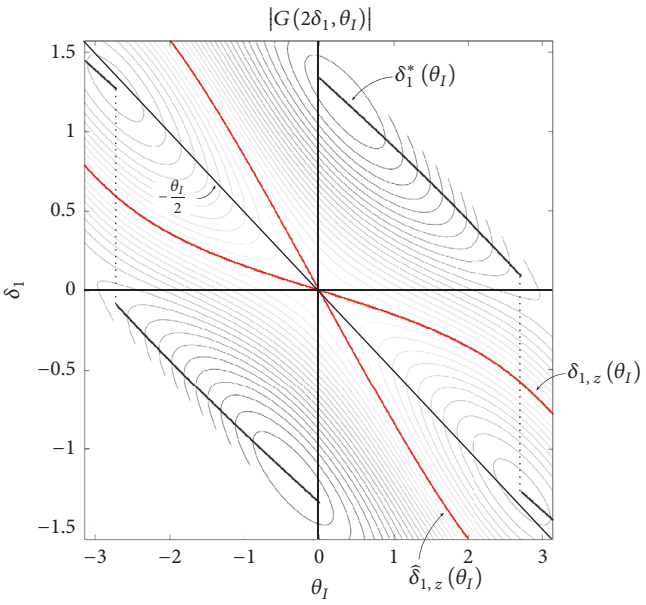


Figure 9: Contour plot of the $|G|$ function.

4.5.3. Implementation Issues. It turns out by numerical examination of $G(2\delta_1, \theta_I)$ that for any θ_I there is exactly one $\delta_{1,z}$ and δ_1^* value; hence $\delta_{1,\max}$ must be unique as well. Figure 9 shows a contour plot of the $|G|$ function, where the thick red lines show the zeros and the thick black lines the maxima of $|G|$. It can be seen that for every θ_I there is another zero $\widehat{\delta}_{1,z}$ outside the interval $[-\theta_I/2, 0]$, but $\delta_{1,z}$ inside the interval is unique.

Unfortunately there are no closed form expressions for calculating $\delta_{1,z}$, δ_1^*, and $\delta_{1,\max}$ values, because G contains Fresnel integrals with δ_1 in the argument. Thus a numerical evaluation is needed which can be done by approximating the Fresnel integrals, for example, by using precalculated lookup tables for $X(\beta)$ and $Y(\beta)$. It is important to note that no two-dimensional lookup table for G is needed because we can decompose it to one-dimensional factors and terms of $X(\beta)$, $Y(\beta)$ and trigonometric functions.

4.5.4. Remarks on Bounded Sharpness. In the above described local planner we payed attention only to the upper bound of the curvature. One could argue that the sharpness should have an upper bound as well to achieve paths which can be traversed by a given minimal velocity. Indeed, as already seen in (15), when a minimal traveling speed is required, then the sharpness of the path should have a given upper bound. The E segments in the EES or $e\bar{e}S$ have a sharpness of $\alpha = \kappa_{CC}^2/2\delta_{CC}$, which has the worst-case value $\alpha = \kappa_{\max}^2/2\delta_{CC}$. It follows that the sharpness is unbounded if δ_{CC} approaches zero, and we can only state that it is less than infinity, because $\delta_{CC} = 0$ means no motion. Hence the curvature is continuous and the maximal allowed speed is greater than zero everywhere.

Let us determine the traveling time along an E segment in case of $|\kappa_{CC}| = \kappa_{\max}$. For this purpose we can use the velocity constraint (14) along the path and the following

identities related to the α, γs, δ, and κ parameters of a clothoid [50]:

$$\gamma s = \frac{2\delta}{\kappa}, \qquad \alpha = \frac{2\delta}{s^2}. \tag{86}$$

Based on these we obtain the length of an E segment (twice as long as a clothoid), the maximal allowed absolute velocity and the minimal travel time along the segment:

$$\begin{aligned} s(\delta_{CC}) &= \frac{4|\delta_{CC}|}{\kappa_{max}}, \\ v_{max}(\delta_{CC}) &= \frac{\sigma_{max}}{|\alpha(\delta_{CC})|}, \\ t_{min}(\delta_{CC}) &= \frac{s(\delta_{CC})}{v_{max}(\delta_{CC})} = \frac{4|\delta_{CC}|}{\kappa_{max}} \cdot \frac{\kappa_{max}^2}{2|\delta_{CC}|\sigma_{max}} \\ &= \frac{2\kappa_{max}}{\sigma_{max}}. \end{aligned} \tag{87}$$

It turns out that the traveling time along such an E segment is finite and lower bounded by $2\kappa_{max}/\sigma_{max}$, independently from δ_{CC}. Thus we can state that the car can go along the planned path in finite time if the number of path segments is finite.

4.6. The TTS Planning Algorithm. At the end of this section, let us summarize how we can use *TTS* paths and the $e\bar{e}S$ steering method in an approximation framework for the purpose of generating collision-free local paths in the presence of obstacles. A mixed algorithm was implemented, which consists of two parts: the first building block is a sampling-based method relying on general *TTS* paths; the second one is the exact $e\bar{e}S$ steering method, as described in Section 4.5.2. We will denote this mixed algorithm as the *TTS-planner* in the sequel. It takes a number of samples from the set of *TTS* paths connecting q_I and q_G and fulfilling the maximal curvature and sharpness constraints. Additionally, it computes the unique $e\bar{e}S$ solution between q_I and q_G as well. From the resulting local path candidates the colliding ones are excluded and the shortest is returned from the remaining set. If every candidate is in collision, then the TTS-planner tries to find new candidates by switching the roles of initial and goal configurations. When no collision-free local path candidate could be found, the TTS-planner reports failure, and the approximation algorithm proceeds with subdividing the global path. The reason of using this mixed local planner approach is twofold. At one hand the $e\bar{e}S$-planner part guarantees convergence of the approximation process, according to its topological property. On the other hand, the sampling part gives a good chance of generating good quality paths by reducing the number of necessary subdivision steps during approximation. In fact, the use of $e\bar{e}S$ steering method alone would be sufficient to ensure convergence, but the sampling part gives a number of alternatives which contribute to path quality and are beneficial when the $e\bar{e}S$ solution collides.

(1) $\mathcal{T}$.Init(q_{init})
(2) **for all** $k = 1$ to K **do**
(3) $q_{rand} \longleftarrow$ RandomConfig()
(4) $q_{near} \longleftarrow \mathcal{T}$.NearestNeighbor($q_{rand}$)
(5) **if** $q_{new} \longleftarrow \mathcal{T}$.Connect($q_{rand}, q_{near}$) **then**
(6) $\mathcal{T}$.AddVertex(q_{new})
(7) $\mathcal{T}$.AddEdge(q_{near}, q_{new})
(8) **end if**
(9) **end for**
(10) **return** $\mathcal{T}$

Algorithm 1: The basic RRT construction algorithm [44].

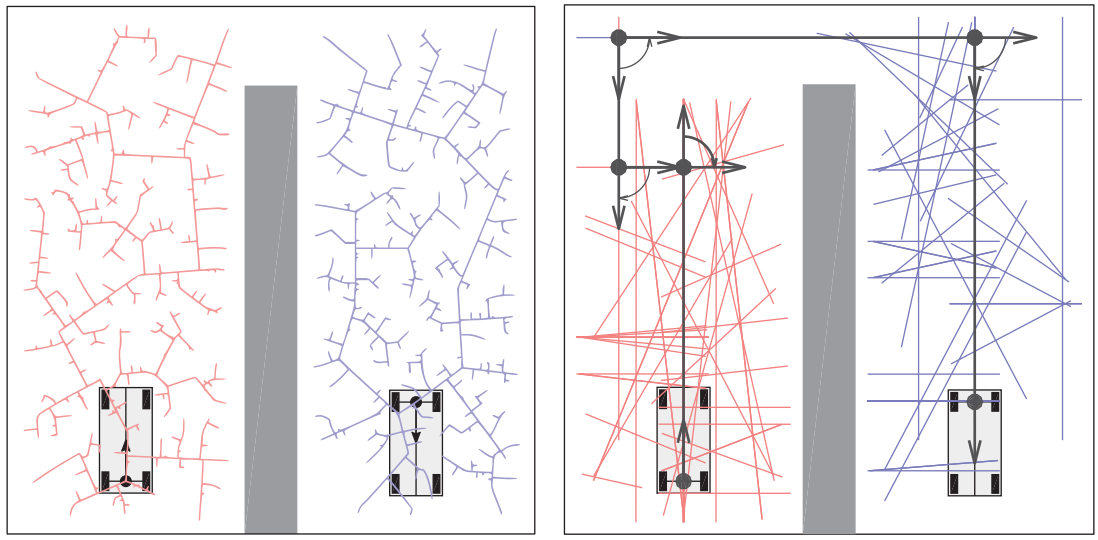
5. Global Planning: The RTR-Planner

In Section 4.4 we argued that "human-like" driving prefers moving along straight path segments. We designed the TTS-planner with this assumption in mind; indeed it tries to find a local path which arrives at the straight line defined by q_G (or q_I). In order to obtain an effective approximation-based planning solution, we need a global planner which facilitates the local planning phase. Thus a preliminary global path is required which is not necessarily feasible for the car, but can be approximated easily with the TTS-planner.

For this purpose we decided not to use any general holonomic planning algorithm; however the topological property of our local planner ensures approximation convergence for any collision-free paths. Instead of this, we looked for a preliminary path which is "almost" feasible for the car. The solution has been found by omitting the minimum turning radius and designing a global planner which uses only straight motion and turning-in-place primitives. The resulting paths are directly applicable for differential drive robots, which have similar motion model as cars except the turning radius constraint. The planning method is based on the Rapidly Exploring Random Trees (RRT) approach [8–10] which is known for its fast convergence properties even in high dimensional configuration spaces and widely used in path planning applications.

Let us first summarize the basic RRT planning process and after that we derive our RTR (rotate-translate-rotate) approach.

5.1. Rapidly Exploring Random Trees. The basic RRT construction process can be seen in Algorithm 1. The building of the tree $\mathcal{T}$ starts from the initial configuration. *RandomConfig()* returns a random configuration q_{rand} from the configuration space $\mathcal{C}$ (sampling step). *NearestNeighbor()* determines the nearest configuration q_{near} in the tree, according to a metric defined on the configuration space (vertex selection step). It depends on the implementation if this function can return only graph vertices or inner configurations of edges as well. *Connect()* tries to connect q_{near} to q_{rand} by simply interpolating between them (tree extension step). It extends q_{near} until it is connected to q_{rand} or a collision is detected. In the latter case q_{new} will be the farthest collision-free configuration towards q_{rand}. In order to reach the goal



(a) (b)

Figure 10: Illustration of RRT and RTR global planning. (a) RRT trees after 1000 iterations and (b) RTR trees and solution path after 65 iterations.

configuration, the random sampling can be biased to include q_{goal} sometimes in the random sequence, or a bidirectional search [10] can be performed by growing two trees from both the initial and goal configurations.

The RRT method can be inherently applied to nonholonomic systems. Instead of a simple interpolation towards q_{rand} (which assumes free mobility in any directions) a system-specific local planner should be applied in the *Connect()* step. The original RRT version proposed in [8] recommended choosing an input and applying for a given time quantum Δt to obtain a Δq which brings q_{near} closer to q_{rand}. This was augmented in [9] for such systems where an explicit local planning (steering) method is available to connect two configurations. Actually we build our approach on this *RRT-Connect* version.

For our purpose (i.e., for differential drive) the simplest steering method is the following sequence of straight motion and turning-in-place primitives: (1) turn to head the next position, (2) move straight until it is reached, and (3) turn to the desired orientation. If a collision occurs during this motion sequence, the extension is stopped at the last collision-free configuration. This simple method has the advantage that it delivers an exact solution between two configurations (no sampling of the input set is needed). Furthermore the tree edges will be straight thus making the nearest neighbor search easy, even if inner configurations of edges are involved in the search. This latter property helps to keep the number of tree vertices as low as possible, although we experienced that a naive application of this rotate-translate-rotate steering method together with RRT has poor performance if narrow areas have to be crossed in order to obtain a result. The reason is that in narrow places the collision will occur most likely during the first rotation primitive; hence no translation, that is, no effective extension of the tree will be achieved. An example scenario with a narrow passage is depicted in Figure 10, with illustrative results of RRT and the below described RTR planners. Both algorithms have been run 100 times; in every run maximum 1000 iterations were allowed. We found that RRT had only 25% success rate and 882.5 iterations on average, while RTR was successful in every trial with averagely 65.4 iterations (see Table 1). We describe the operation of RTR-planner in the sequel.

Table 1: Performance measures of RRT and RTR (based on 100 runs).

	RRT	RTR
Success Ratio	25%	100%
Avg. no. of iterations	882.5	65.4

5.2. RT-Trees and Primitives. RTR-planner is similar to a bidirectional RRT-Connect algorithm; however, it has differences in the sampling, the vertex selection, and the extension steps as well. It uses translation (T) and rotation (R) primitives for building the trees. In the extension steps, only RT (rotate-translate) sequences are used instead of the exact rotate-translate-rotate steering method. For this reason, we denote the constructed trees as RT-trees. The vertices of the trees are configurations as usual and the edges are continua of configurations. According to T and R motion primitives, these are called Translational Configuration Intervals (TCIs) and Rotational Configuration Intervals (RCIs). The process of an RT-tree construction can be seen in Algorithm 2 and is detailed in the sequel.

5.2.1. Sampling. The first difference to RRT can be found in the sampling step. *RandomPos()* returns a random position p_G without orientation, denoted as the guiding position in the sequel, instead of a configuration. It can be treated as

```
(1)  function RT_CONSTRUCT(q_init)
(2)    T. Init(q_init)
(3)    for all k = 1 to K do
(4)      p_G ⟵ RandomPos()
(5)      q_near ⟵ T. NearestNeighbor(p_G)
(6)      turnDir ⟵ MinTurnDir(q_near, p_G)
(7)      collision ⟵ T. Extend(q_near, turnDir)
(8)      if collision then
(9)        T. Extend(q_near, −turnDir)
(10)     end if
(11)   end for
(12) return T
(13) end function
(14)
(15) function T. Init(q)
(16)   T. AddVertex(q)
(17)   TCI ⟵ TCIExtend (q, "forward")
(18)   T. AddVertex(TCI.q_end)
(19)   T. AddEdge(q, TCI)
(20)   TCI ⟵ TCIExtend (q, "backward")
(21)   T. AddVertex(TCI.q_end)
(22)   T. AddEdge(q, TCI)
(23) end function
(24)
(25) function T. Extend(q, turnDir)
(26)   [RCI, collision] ⟵ RCIExtend(q, turnDir)
(27)   T. AddVertex(RCI.q_end)
(28)   T. AddEdge(q, RCI)
(29)   TCI ⟵ TCIExtend (RCI.q_end, "forward")
(30)   T. AddVertex(TCI.q_end)
(31)   T. AddEdge(RCI.q_end, TCI)
(32)   TCI ⟵ TCIExtend(RCI.q_end, "backward")
(33)   T. AddVertex(TCI.q_end)
(34)   T. AddEdge(RCI.q_end, TCI)
(35) return collision
(36) end function
```

Algorithm 2: Construction of an RT-tree.

a one-dimensional continuous set of configurations, from which any element can serve as local goal in the tree extension step.

5.2.2. Vertex Selection. $\mathcal{T}$*.NearestNeighbor()* returns the configuration in the existing tree which has the smallest *position* distance to p_G. This step uses a simple Euclidean metric; hence no special configuration space metrics are needed.

5.2.3. Extension. The main difference to the RRT method can be found in the tree extension step. On the one hand, translation is always performed in both forward and backward directions. Additionally, the translation is not stopped when p_G is reached, but continued until the first collision in both directions (this is the functionality of *TCIExtend()*, called by both $\mathcal{T}$*.Init()* and $\mathcal{T}$*.Extend()* functions). On the other hand, an important difference is the fact that an R-T sequence will always be planned, even if a collision occurs in the rotation phase. The RTR-planner balances between reaching guiding positions and extending the tree. If p_G cannot be reached due to a collision, then two things can be done:

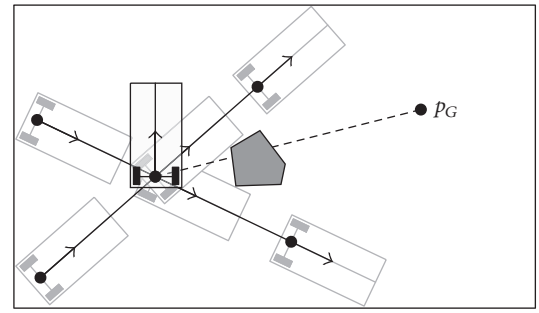


Figure 11: Illustration of the tree extension procedure if the direction to p_G is blocked.

(1) If the collision occurred during the T primitive, then the iteration is finished (because the tree has been extended).

(2) If the collision occurred during the R primitive, then TCI_EXTEND is performed in both forward and backward directions at the colliding orientation (first extension), and the rotation is tried again in the other turning direction as well. After the second rotation, independently from its success or collision, TCI_EXTEND is called again (second extension). An example of this procedure can be seen in Figure 11.

This extension strategy causes a more aggressive free space exploration than in case of the basic RRT method.

5.3. Connecting RT-Trees. As the RTR-planner builds two RT-trees—from both the initial and goal configurations—in order to obtain a final path, the two trees have to be connected. This is attempted in every iteration. The newly added TCIs are checked against every TCI in the other tree, starting from its root. If an intersection is found and if a collision-free RCI can be put between the intersecting TCIs to connect them, then the two trees can be merged and the path determined easily by tracing the trees back to their roots.

6. Simulation Results

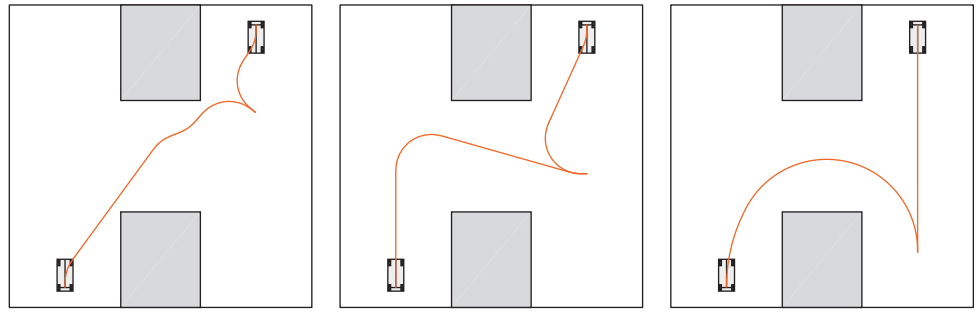
The RTR + TTS planning algorithm was tested extensively in simulations and compared to other similar approaches in terms of effectiveness and path quality. To two other algorithms were investigated:

(i) *RRT-Connect (CCRS).* Bidirectional RRT using the CCRS steering method as local planner (direct one-step approach, no approximation needed).

(ii) *RTR + CCRS.* The RTR-planner is used for generating a preliminary global path, followed by an approximation phase using CCRS paths.

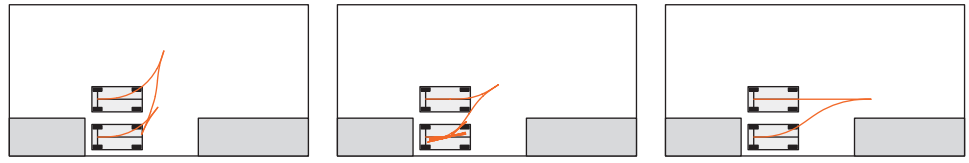
The following scenarios have been used for testing:

(i) A wide environment with few obstacles (can be treated as an easy planning task, see Figure 12).



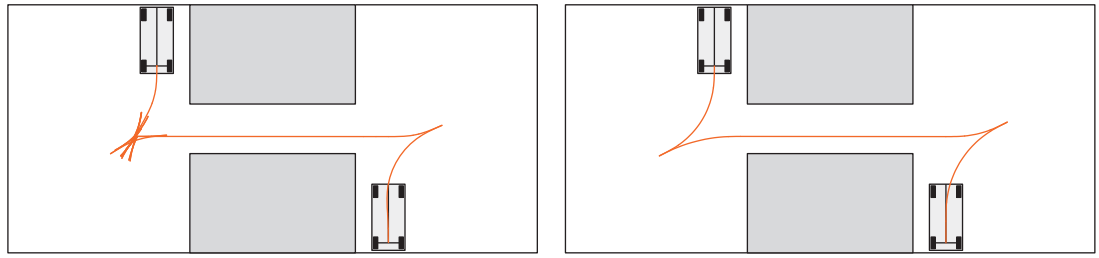
(a) RRT-Connect (CCRS) (b) RTR + CCRS (c) RTR + TTS (proposed method)

Figure 12: Path planning in a wide environment.



(a) RRT-Connect (CCRS) (b) RTR + CCRS (c) RTR + TTS (proposed method)

Figure 13: Path planning for parallel parking.



(a) RTR + CCRS (b) RTR + TTS (proposed method)

Figure 14: Crossing a narrow corridor.

(ii) A simple parallel parking scenario (a well known problem for path planning algorithms, see Figure 13).

(iii) Driving through a narrow corridor and parking next to the wall (can be treated as a difficult planning task, see Figure 14).

(iv) Crossing three perpendicular narrow corridors with bigger free areas at the corners (can be treated as a very difficult planning task, see Figure 15).

The simulated car is 4 m long and 2 m wide with 4.42 m minimal turning radius.

To obtain practical results, we constrained the maximum iteration count of the tree building process to 10000 in every algorithm and applied a lower bound of path subdivision at the approximation phase. Every algorithm has been run 100 times in every scenario, and the average results are shown in Figure 16. The performance and path quality is measured by the following metrics:

(i) *Success Ratio.* Percentage of successful runs.

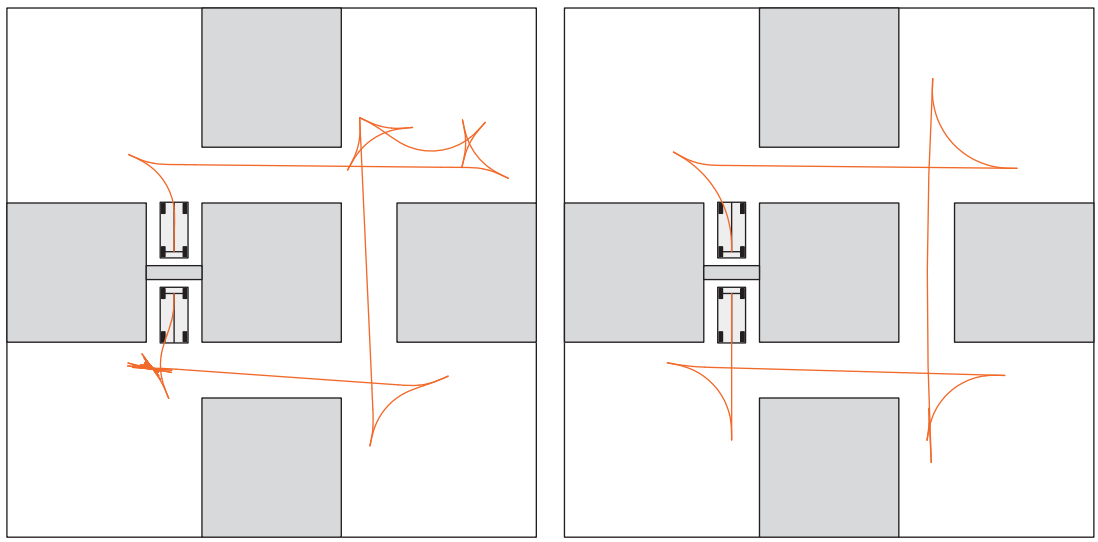
(ii) *Number of Cusps.* The number of reversals along the path. A good path contains as few reversals as possible.

(iii) *Steering Amount.* The absolute integral of the path curvature. It gives the amount of turning along the path. A path with less sharp turns or with more straight segments is more comfortable than a path requiring much turning along the way.

(iv) *Travel Time.* The time of driving along the resulting path is estimated as follows. A traveling speed function with $v_{max} = 5$ m/s and $v_{min} = 1$ m/s is assigned to the path, which is inversely proportional to the actual path curvature. The cusps are penalized with 0.5 s "reversing time." Those paths are better, which require less time to travel along.

Numerical results are listed in Figure 16, while illustrative examples can be examined in Figures 12–15.

As expected, the "easy" path planning task in wide environment can be flawlessly solved by all selected algorithm



(a) RTR + CCRS (b) RTR + TTS (proposed method)

Figure 15: Crossing three narrow corridors.

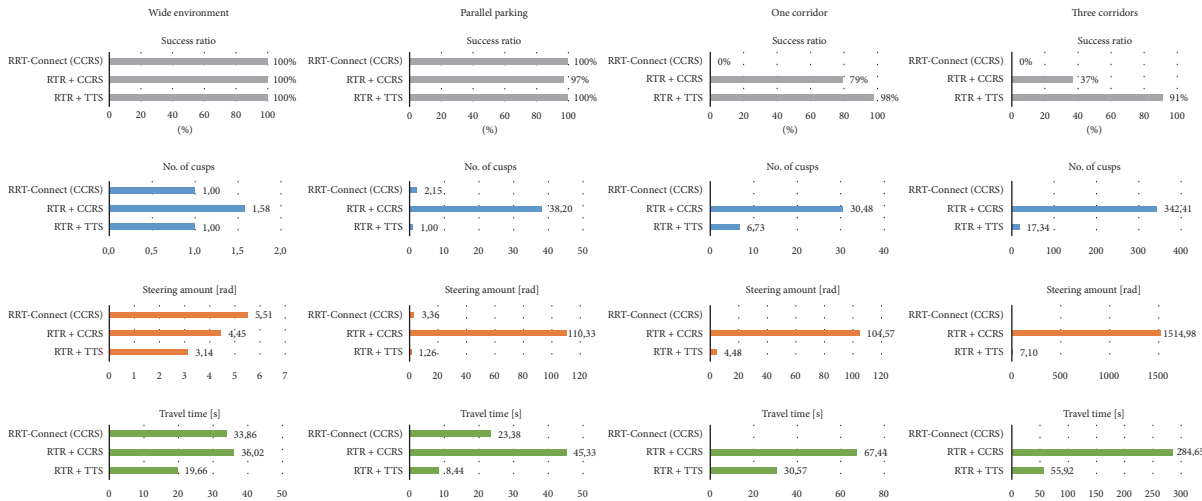


Figure 16: Summary of the simulation results.

combinations (Figure 12). The path quality measures are similar (first column in Figure 16); however the proposed RTR + TTS-planner combination results in the most comfortable and human-like paths with respect to steering amount and travel time.

Similarly, the parallel parking task is solvable as well by every tested algorithm (Figure 13). The TTS-based local planner obtains the highest quality paths. This is not surprising because the most "natural" parallel parking maneuver is a *TTS* path itself. The reason for obtaining more complicated paths with CCRS-based methods is that the CCRS planner does not have the "natural" solution amongst its local path candidates; hence it relies stronger on the global planning phase. During parallel parking we want to achieve a lateral shift to a narrow place which is quite unnatural for the car; this results in many approximation steps for the RTR + CCRS planner, producing an enormous number of small movements. The result is much better if the CCRS local planner is applied directly in the RRT-Connect planning phase, because in this case the unnatural approximation is avoided. However, the RTR + TTS-planner outperforms this as well (see the second column in Figure 16).

As we move to more complicated planning challenges in environments containing narrow corridors, sampling-based planners need more iterations to find a solution, resulting in lower success rates (because the number of iterations is limited). Those global planners which tend to use mainly straight movements are more successful than others. In the scenario with one corridor the direct RRT-Connect method completely fails since it uses curved local paths to connect configuration samples (for this reason only RTR + CCRS and RTR + TTS solutions are depicted in

Figure 14). Approximation methods are better suited to such environments and it can be seen in the third column of 14 that the TTS local planner results in higher success ratios and better quality paths. The effectiveness of RTR as global planner originates from the aggressive extension step and from the fact that it contains only straight forward and backward translation movements. Because *TTS* paths are designed to arrive at straight lines, they are well suited for the approximation of RTR paths, resulting in less subdivision steps (efficiency) and simpler path geometry (quality).

The effectiveness of the RTR global planner can mostly be examined in the last scenario, where three narrow corridors have to be passed (Figure 15). Comparing the approximation result using CCRS and TTS local planners, we can find a much higher efficiency and much better path quality in case of TTS. Furthermore, we can see the different philosophies behind the two local planners; namely, CCRS paths strive to have the maximal allowed curvature, while *TTS* paths are not restricted to this. In the larger "turning spaces" between corridors, the TTS-planner generates less sharp turns and needs less reversals, which increases the comfort of passengers in the car. The direct RRT-Connect method with CCRS local planner completely fails, similarly to the previous example. As can be seen in the last column of Figure 16, the RTR + TTS-planner significantly outperforms the other algorithms.

7. Conclusions

In this paper we presented a global planning method for car-like vehicles, producing paths with continuous curvature. A comparative analysis with possible alternatives was performed as well and we could see that both components (RTR and TTS) of the two-phase planning approach contribute to the effectiveness and path quality of the resulting algorithm. The (preliminary) RTR path planner is capable of designing paths consisting of straight movements and turning-in-place. We apply the TTS local planner to the RTR path in a second approximation phase, in order to obtain a final path obeying the bounded and continuous curvature conditions. The TTS-planner uses straight segments, CC-turns, and elementary paths in order to generate a feasible and human-like solution even in cluttered and narrow environments. We gave an overview of how to build CC-turns and elementary paths; the class of curvature and sharpness bounded *EES* and *TTS* paths were introduced, and existence conditions were given for them. An exact steering method: the $e\bar{e}S$-planner was presented as well, which delivers a unique *EES* solution for a planning query and verifies the topological property. Simulation studies were presented to illustrate that the proposed RTR + TTS planning algorithm can be applied universally in less and more difficult situations and the obtained paths are quite "natural."

We continue this work in the future by implementing the proposed algorithm on our physical testbed [52, 53] utilizing small-sized model cars and later on a real vehicle to perform real-life tests and performance measurements. Further research includes the investigation of possible applications in previously unknown and changing environments, and the interaction possibilities of this planning method with online map building algorithms.

Appendix

A. Existence Conditions for *EES* Paths

Lemma 2 stated that, in the absence of obstacles, any (q_I, q_G) pair of initial and goal configurations of a continuous steering car can be connected by an *EES* path. Here we give its proof, where we use some properties of $B(\cdot)$ and $D(\cdot,\cdot)$ functions, which are established in Appendix C.

Proof of Lemma 2. Let us apply the curvature bound to κ_2 first:

$$\left|\kappa_2\right| \leq \kappa_{\max}. \tag{A.1}$$

After substituting (59b) it gets the form

$$\left|\kappa_2\right| = \left|\frac{B\left(\widetilde{\theta}_I\right)}{\widetilde{y}_I}\right| \leq \kappa_{\max}. \tag{A.2}$$

Using the fact that $B(\cdot)$ is nonnegative in the interval of interest (cf. Property C.3 in Appendix C) we can rearrange this to

$$\left|\widetilde{y}_I\right| \geq \frac{B\left(\widetilde{\theta}_I\right)}{\kappa_{\max}} = \left|\Delta y_{2,\min}\left(\widetilde{\theta}_I\right)\right|, \tag{A.3}$$

which means that, for a given $\widetilde{\theta}_I = -2\delta_2$, a valid *ES* subpath exists if and only if $\widetilde{q}_I$ is farther from the x-axis than the minimal possible change in the y-coordinate determined by the necessary deflection and the given curvature constraint.

We can obtain the necessary and sufficient conditions for the existence of *EES* paths by substituting (55a), (55b), (55c), and (55d) into (A.3):

$$\left|y_I + \frac{D\left(2\delta_1, \theta_I\right)}{\kappa_1}\right| \geq \frac{B\left(2\delta_1 + \theta_I\right)}{\kappa_{\max}}, \tag{A.4}$$

alternatively written as

$$\left|y_I + \Delta y_1\left(\theta_I, \delta_1, \kappa_1\right)\right| \geq \left|\Delta y_{2,\min}\left(\theta_I, \delta_1\right)\right|, \tag{A.5}$$

together with the obvious other condition $|\kappa_1| \leq \kappa_{\max}$.

According to Property C.5 in Appendix C, $D(2\delta_1, \theta_I) = 0$ only if $\delta_1 = 0$ or $\delta_1 = -\theta_I \pm \pi$. This means that, for any θ_I and $\delta_1 \notin \{0, -\theta_I \pm \pi\}$, the Δy_1 term is nonzero and a κ_1 can be found such that (A.5) and the curvature bound are fulfilled, because

$$\lim_{\kappa_1 \to 0}\left|\frac{D\left(2\delta_1, \theta_I\right)}{\kappa_1}\right| = \infty. \tag{A.6}$$

We have seen that for any query pair of the form $(q_I', 0)$ a class of *EES* paths—parametrized by (δ_1, κ_1)—is available. A more general query pair (q_I, q_G) can be transformed to the form $(q_I', 0)$ by the inverse transformation $T^{-1}(q_G)$, and the resulting path can be transformed back by $T(q_G)$. □

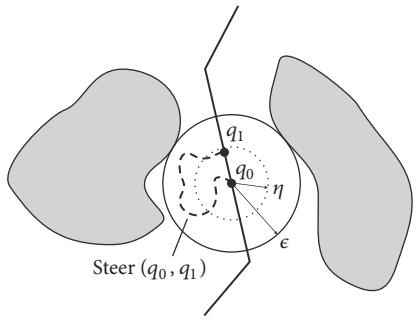


FIGURE 17: Illustration of topological property for $\mathscr{C} = \mathbb{R}^2$.

B. Topological Property of the $e\bar{e}S$ Steering Method

The topological property of steering methods is defined as follows [12].

Definition B.1 (topological property). A steering method Steer(q_0, q_1) verifies the topological property if

$$\begin{aligned} &\forall \epsilon > 0, \\ &\exists \eta > 0, \\ &\forall q_0, q_1 \in \mathscr{C} \\ &d_{\mathscr{C}}(q_0, q_1) < \eta \Longrightarrow d_{\mathscr{C}}(q_0, \text{Steer}(q_0, q_1)(s)) < \epsilon, \\ &\qquad\qquad \forall s \in [0, S_\lambda], \end{aligned} \tag{B.1}$$

where $d_{\mathscr{C}}$ is any metric defined on the configuration space $\mathscr{C}$ and S_λ is the total length of the of the path returned by Steer(q_0, q_1).

This means that, for every nonzero ϵ, η can be found such that if q_1 is in an η-neighborhood of q_0, then the local path generated by Steer does not exit the ϵ-neighborhood of q_0. When approximating a global path and having q_0 as an intermediate point on it which has a collision-free ϵ-neighborhood, then we can find another intermediate point q_1 along the path in the η-neighborhood of q_0 such that the feasible local path Steer(q_0, q_1) is collision-free (see Figure 17). Hence if the global path has nonzero clearance, then the topological property ensures that a feasible path can be constructed along it using a sequence of local paths generated by the steering method.

We will show in the sequel that the $e\bar{e}S$ steering method has this advantageous property. But before that, a metric $d_{\mathscr{C}}$ has to be defined over $\mathscr{C}$. The most important metrics over the n-dimensional Euclidean space $\mathbb{R}^n$ belong to the family of L_p metrics [6]:

$$L_p(r, r') = \left(\sum_{i=1}^{n} \left| r_i - r_i' \right|^p \right)^{1/p}, \quad r, r' \in \mathbb{R}^n. \tag{B.2}$$

A special L_p metric is the L_∞ metric which can be obtained from (B.2) if $p \to \infty$:

$$L_\infty(r, r') = \max_{1 \le i \le n} \left\{ \left| r_i - r_i' \right| \right\}. \tag{B.3}$$

The configuration space of the continuous steering car is not an Euclidean space but a manifold $\mathscr{C} = \mathbb{R}^2 \times \mathbb{S}^1 \times [-\kappa_{\max}, \kappa_{\max}]$, where $\mathbb{S}^1$ stands for the unit circle which is homeomorphic to $[-\pi, \pi]/\sim$. Because of the identification of $-\pi$ and π, special care must be taken when the "distance" of two angles has to be measured. An appropriate metric for $\mathbb{S}^1$ can be the following [6]:

$$d_{\mathbb{S}^1}(\theta, \theta') = \min \left\{ \left| \theta - \theta' \right|, 2\pi - \left| \theta - \theta' \right| \right\}. \tag{B.4}$$

As stated in [6], metric spaces are extendable by Cartesian product and a new metric can be defined for the extended space using the original metrics. Since our configuration space $\mathscr{C}$ is an extension over $\mathbb{R}^2$ and $\mathbb{S}^1$, we can choose L_∞ metric for $\mathbb{R}^2$, $d_{\mathbb{S}^1}$ for $\mathbb{S}^1$, and L_∞ again over these:

$$\begin{aligned} &d_{\mathscr{C}}(q_0, q_1) \\ &\quad = L_\infty \left(L_\infty \left((x_0, y_0), (x_1, y_1) \right), d_{\mathbb{S}^1}(\theta_0, \theta_1) \right), \end{aligned} \tag{B.5}$$

which leads to

$$\begin{aligned} d_{\mathscr{C}}(q_0, q_1) = \max \{ &\left| x_0 - x_1 \right|, \left| y_0 - y_1 \right|, \\ &\min \left\{ \left| \theta_0 - \theta_1 \right|, 2\pi - \left| \theta_0 - \theta_1 \right| \right\} \}. \end{aligned} \tag{B.6}$$

Note that we have neglected one dimension of the configuration space (the $[-\kappa_{\max}, \kappa_{\max}]$ part) in the metric. This is valid because the curvatures of q_I and q_G are always 0; thus the "curvature distance" of these is zero as well. Furthermore, note that in this metric Euclidean distances are compared to angles, which is a common problem in motion planning. It is hard to define a metric over $\mathscr{C}$ that can avoid this issue. Nevertheless, the above defined metric is appropriate for proving the following theorem.

Theorem B.2. *The $e\bar{e}S$ steering method verifies the topological property.*

Proof. Since the goal configuration is at the origin by assumption (without loss of generality), the metric (B.6) can be written in the following form:

$$d_{\mathscr{C}}(q(s)) = \max \{ |x(s)|, |y(s)|, |\theta(s)| \}, \tag{B.7}$$

where $q(s)$ is a shorthand for Steer $(0, q_I)(s)$. The total length of an $e\bar{e}S$ path is

$$S_\lambda = s_1 + s_2 + s_3 = \left| \frac{4\delta_1}{\kappa} \right| + \left| \frac{4\delta_2}{\kappa} \right| + s_3, \tag{B.8}$$

where s_1, s_2, and s_3 are the corresponding path segment lengths, according to (86). It is obvious that

$$|x(s)| \le s \le S_\lambda, \tag{B.9}$$

$$|y(s)| \le s \le S_\lambda, \tag{B.10}$$

$$|\theta(s)| \le |2\delta_1| + |2\delta_2| \tag{B.11}$$

for all $s \in [0, S_\lambda]$. After multiplying (B.11) with $2/|\kappa|$ we get

$$\frac{2\,|\theta(s)|}{|\kappa|} \leq s_1 + s_2 \leq S_\lambda, \tag{B.12}$$

which results in

$$|\theta(s)| \leq S_\lambda \frac{|\kappa|}{2} \leq S_\lambda \frac{\kappa_{\max}}{2}. \tag{B.13}$$

Looking at (B.7) and (B.9)–(B.13) we can state that

$$d_{\mathscr{C}}(q(s)) \leq S_\lambda \cdot \max\left\{1, \frac{\kappa_{\max}}{2}\right\}. \tag{B.14}$$

With this, the statement to be proven is equivalent to

$$d_{\mathscr{C}}(q_I) < \eta \Longrightarrow S_\lambda \cdot \max\left\{1, \frac{\kappa_{\max}}{2}\right\} < \epsilon, \tag{B.15}$$

which holds when

$$\lim_{q_I \to 0} S_\lambda = 0. \tag{B.16}$$

According to (83), $q_I \to 0$ means that $|\kappa^*| \to \infty$; hence $|\kappa| \to \kappa_{\max}$. It follows that

$$\lim_{q_I \to 0} S_\lambda = \frac{|4\delta_1|}{\kappa_{\max}} + \frac{|4\delta_2|}{\kappa_{\max}} + s_3. \tag{B.17}$$

Let us define the function

$$F(y_I, \theta_I, \delta_1) := y_I + \frac{G(2\delta_1, \theta_I)}{\kappa} \tag{B.18}$$

for any fixed κ. According to (77b) we know that for any $e\bar{e}S$ path

$$F(y_I, \theta_I, \delta_1) = 0. \tag{B.19}$$

Let us examine the behavior of $F(\cdot)$ in the vicinity of $(y_I, \theta_I) = (0, 0)$:

$$\begin{aligned}\lim_{(y_I,\theta_I)\to(0,0)} F(y_I, \theta_I, \delta_1) &= 0 + \frac{G(2\delta_1, 0)}{\pm\kappa_{\max}} \\ &= \pm\frac{2B(2\delta_1)}{\kappa_{\max}},\end{aligned} \tag{B.20}$$

where we applied Property C.4 in Appendix C. This implies together with (B.19)

$$\pm\frac{2B(2\delta_1)}{\kappa_{\max}} = 0. \tag{B.21}$$

Property C.3 in Appendix C states that

$$B(2\delta_1) = 0 \Longleftrightarrow \delta_1 = 0, \tag{B.22}$$

which implies

$$\lim_{(y_I,\theta_I)\to(0,0)} F(y_I, \theta_I, \delta_1) = 0 \Longleftrightarrow \delta_1 \longrightarrow 0. \tag{B.23}$$

In other words, in the neighborhood of $(y_I, \theta_I) = (0, 0)$ an $e\bar{e}S$ path can only exist iff $\delta_1 \to 0$.

To sum it up, we now that if $q_I \to 0$, then

$$\begin{aligned}|\kappa| &\longrightarrow \kappa_{\max}, \\ \delta_1 &\longrightarrow 0, \\ \delta_2 &= -\delta_1 - \frac{\theta_I}{2} \longrightarrow 0.\end{aligned} \tag{B.24}$$

Furthermore, based on (77a) we get

$$\lim_{q_I \to 0} s_3 = \lim_{q_I \to 0} \left| \cancelto{0}{x_I} + \frac{\cancelto{0}{A(2\delta_1 + \theta_I)} + \cancelto{0}{C(2\delta_1, \theta_I)}}{\cancel{\kappa}\,\kappa_{\max}} \right| = 0, \tag{B.25}$$

where we used (B.23) and Property C.1 in Appendix C. Finally, we can conclude that

$$\lim_{q_I \to 0} S_\lambda = \cancelto{0}{\frac{|4\delta_1|}{\kappa_{\max}}} + \cancelto{0}{\frac{|4\delta_2|}{\kappa_{\max}}} + \cancelto{0}{s_3} = 0, \tag{B.26}$$

which proves (B.15); thus the requirement for the topological property is fulfilled. □

C. Auxiliary Functions

Some auxiliary functions are used during deriving the *EES* and $e\bar{e}S$ paths:

$$X(\beta) = \operatorname{sgn}(\beta)\sqrt{\pi|\beta|} \cdot C_F\left(\sqrt{\frac{|\beta|}{\pi}}\right) \tag{C.1}$$

$$Y(\beta) = \sqrt{\pi|\beta|} \cdot S_F\left(\sqrt{\frac{|\beta|}{\pi}}\right) \tag{C.2}$$

$$A(\beta) = X(\beta)(1 + \cos\beta) + Y(\beta)\sin\beta \tag{C.3}$$

$$B(\beta) = X(\beta)\sin\beta + Y(\beta)(1 - \cos\beta) \tag{C.4}$$

$$C(\beta, \psi) = A(\beta)\cos\psi - B(\beta)\sin\psi \tag{C.5}$$

$$D(\beta, \psi) = A(\beta)\sin\psi + B(\beta)\cos\psi \tag{C.6}$$

$$G(\beta, \psi) = B(\beta + \psi) + D(\beta, \psi). \tag{C.7}$$

Because all of these contain Fresnel integrals, it is hard to solve equations including these functions if the unknown variable is in the argument. However, some function properties can be specified including parity, zeros, and specific values. These properties are useful when dealing with expressions containing them.

Property C.1. The functions $X(\beta)$, $Y(\beta)$, $A(\beta)$, $B(\beta)$, $C(\beta, \psi)$, and $D(\beta, \psi)$ have zero value at $\beta = 0$.

Proof. For $X(\beta)$ and $Y(\beta)$ the value at $\beta = 0$ can be obtained by substitution: all factors are zero. It follows from the definitions that $X(0)$ and $Y(0)$ make every term zero in the other functions as well. □

Table 2: Signs of the terms of $B(\beta)$.

β	$X(\beta)$	$\sin\beta$	$Y(\beta)$	$1-\cos\beta$	$B(\beta)$
$-\pi$	$-$	0	$+$	$+$	$+$
$(-\pi,0)$	$-$	$-$	$+$	$+$	$+$
0	0	0	0	0	0
$(0,\pi)$	$+$	$+$	$+$	$+$	$+$
π	$+$	0	$+$	$+$	$+$

Property C.2. $X(\beta)$ and $A(\beta)$ are odd, while $Y(\beta)$ and $B(\beta)$ are even functions. Additionally, $\operatorname{sgn}(X(\beta)) = \operatorname{sgn}(\beta)$ and $Y(\beta) \geq 0$.

Proof. It follows from the definition of Fresnel sine and cosine integrals (18a) and (18b) that they are odd functions and their signs are the same as the sign of their arguments. In the definitions of $X(\beta)$ and $Y(\beta)$, the arguments of C_F and S_F depend on the absolute value of β, which results in even functions with nonnegative values. The multiplication factor $\sqrt{\pi|\beta|}$ is even as well, while the $\operatorname{sgn}(\cdot)$ function is odd. We know that the product of two functions is even if the parities of factors are the same. Similarly, if the parities differ, then the resulting function is odd. Based on these, it is obvious that $X(\beta)$ is odd with $\operatorname{sgn}(X(\beta)) = \operatorname{sgn}(\beta)$ and $Y(\beta)$ is even and nonnegative.

Given that the sine function is odd and cosine is even and $X(0) = Y(0) = 0$, it can be easily seen that both terms in the definition of $A(\beta)$ are odd; thus $A(\beta)$ itself is odd as well. Similarly, both terms in $B(\beta)$ are even; thus $B(\beta)$ itself is even as well. □

Property C.3. In the range $\beta \in [-\pi, \pi]$, $B(\beta)$ is positive except at $\beta = 0$, where it has zero value.

Proof. Let us see the sign of terms of $B(\beta)$ in different subsets of range $\beta \in [-\pi, \pi]$, as listed in Table 2. It can be seen that $B(\beta)$ is a sum of positive or zero-valued terms. Only $X(\beta)$ and $\sin\beta$ can be negative, but in all such cases the product of them is positive or zero. The table shows that

$$B(\beta) = 0 \Longleftrightarrow \beta = 0, \tag{C.8}$$

$$B(\beta) > 0 \Longleftrightarrow \beta \neq 0. \tag{C.9}$$

□

Property C.4. $D(\beta, \psi)$ has the following specific values:

$$D(\beta, 0) = B(\beta) \tag{C.10}$$

$$D(-\beta, \beta) = D(\beta, -\beta) = -B(\beta). \tag{C.11}$$

Proof. (C.10) can be verified easily by substituting $\psi = 0$ into (C.4). Before looking at the second equation, let us reformulate the definition of $D(\beta)$ as follows:

$$\begin{aligned} D(\beta, \psi) = {} & X(\beta)(\sin\psi + \sin(\beta+\psi)) \\ & + Y(\beta)(\cos\psi - \cos(\beta+\psi)). \end{aligned} \tag{C.12}$$

We used here the definitions of $A(\beta)$ and $B(\beta)$ and trigonometric addition rules. In both cases of (C.11) we get

$$\begin{aligned} D(-\beta, \beta) &= D(\beta, -\beta) \\ &= -X(\beta)\sin\beta + Y(\beta)(\cos\beta - 1), \end{aligned} \tag{C.13}$$

which is obviously equal to $-B(\beta)$. □

Property C.5. The function $D(\beta, \psi)$ has zeros only at $\beta = 0$ and $\beta = -2\psi + 2\pi k$, $k \in \mathbb{Z}$ in the range $\beta \in [-\pi, \pi]$.

Proof. The first condition for $D(\beta, \psi)$ being zero was already stated by Property C.1. The second condition can be verified easily by substitution to (C.12). Furthermore, we have to check whether $D(\beta, \psi)$ has zeros elsewhere. Let us assume that $\beta \neq 0$ and $\beta \neq -2\psi + 2\pi k$, but $D(\beta, \psi) = 0$. In this case the following equality would be true:

$$\begin{aligned} & X(\beta)(\sin\psi + \sin(\beta+\psi)) \\ & \quad = -Y(\beta)(\cos\psi - \cos(\beta+\psi)). \end{aligned} \tag{C.14}$$

Let us introduce a new variable

$$\alpha := \beta + 2\psi - 2\pi k \neq 0 \tag{C.15}$$

according to our assumption $\beta \neq -2\psi + 2\pi k$. After substituting $\psi = \alpha/2 - \beta/2 + \pi k$ into (C.14) we get

$$\begin{aligned} & X(\beta)\left[\sin\left(\frac{\alpha}{2} - \frac{\beta}{2} + \pi k\right) + \sin\left(\frac{\alpha}{2} + \frac{\beta}{2} + \pi k\right)\right] \\ & \quad = -Y(\beta) \\ & \quad \cdot \left[\cos\left(\frac{\alpha}{2} - \frac{\beta}{2} + \pi k\right) - \cos\left(\frac{\alpha}{2} + \frac{\beta}{2} + \pi k\right)\right]. \end{aligned} \tag{C.16}$$

By applying trigonometric addition rules we can reformulate it:

$$\begin{aligned} & X(\beta) \cdot \cancel{2\sin\left(\frac{\alpha}{2} + \pi k\right)} \cdot \cos\frac{\beta}{2} \\ & \quad = -Y(\beta) \cdot \cancel{2\sin\left(\frac{\alpha}{2} + \pi k\right)} \cdot \sin\frac{\beta}{2}. \end{aligned} \tag{C.17}$$

The $2\sin(\alpha/2 + \pi k)$ terms can be canceled because they are nonzero according to (C.15). The equality should hold for the signs as well:

$$\begin{aligned} & \operatorname{sgn}(X(\beta))\operatorname{sgn}\left(\cos\frac{\beta}{2}\right) \\ & \quad = -\operatorname{sgn}(Y(\beta))\operatorname{sgn}\left(\sin\frac{\beta}{2}\right), \end{aligned} \tag{C.18}$$

which evaluates to

$$\begin{aligned} \operatorname{sgn}(\beta) &= -\operatorname{sgn}(\beta), \quad \text{if } \beta \in (-\pi, \pi) \setminus 0, \\ 0 &= 1, \quad \text{if } \beta = -\pi, \\ 0 &= -1, \quad \text{if } \beta = \pi. \end{aligned} \tag{C.19}$$

Because all of these are contradictions, the assumption that $D(\beta, \psi)$ has zeros elsewhere than $\beta = 0$ and $\beta = -2\psi + 2\pi k$ was wrong. □

Property C.6. In the range $\beta \in [-\pi, \pi]$ and $\psi \in [-\pi, \pi]$, for all ψ there exists β such that $G(\beta, \psi) \neq 0$.

Proof. First let us see the case $\psi = 0$. Using (C.7) and (C.10) we get

$$G(\beta, 0) = 2B(\beta). \tag{C.20}$$

According to (C.9) we can state that $G(\beta \neq 0, \psi = 0) \neq 0$.

We can check this for $\psi \neq 0$ as well. For example, let us choose $\psi = -\beta$. Using (C.7) and (C.11) we arrive at

$$G(\beta, -\beta) = -B(\beta). \tag{C.21}$$

Because $\beta = -\psi \neq 0$, based on (C.9) we can state again that in case of $\psi \neq 0$ a β can be found such that $G(\beta, \psi) \neq 0$. □

Property C.7. In the range $\beta \in [-\pi, \pi]$ and $\psi \in [-\pi, \pi]$, for all ψ there exists β such that $G(\beta, \psi) = 0$.

Proof. Let us look again at the case $\psi = 0$ first. Based on (C.20) and (C.8) we can state that $G(0, 0) = 0$.

In the case $\psi \neq 0$ let us examine two specific values for β:

(i) If $\beta = 0$, then, using (C.7) and Properties C.1 and C.3, we get

$$G(0, \psi) = B(\psi) > 0. \tag{C.22}$$

(ii) If $\beta = -\psi$, then, using (C.7) and (C.11) and Property C.3, we get

$$G(0, \psi) = -B(\psi) < 0. \tag{C.23}$$

It follows that for all ψ there exists a $\beta \in (-\psi, 0)$ such that $G(\beta, \psi) = 0$. □

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is connected to the scientific program of the "Development of Quality-Oriented and Harmonized R + D + I Strategy and Functional Model at BME" project. This project is supported by the New Széchényi Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002).

References

[1] J. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, Mass, USA, 1991.

[2] J.-P. Laumond, *Robot motion planning and control*, vol. 229 of *Lecture Notes in Control and Information Sciences*, Springer, London, UK, 1998.

[3] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006, http://planning.cs.uiuc.edu/.

[4] J. Rigelsford, "Introduction to Autonomous Mobile Robots20043R. Siegwart and I.R. Nourbakhsh. ," *Industrial Robot: An International Journal*, vol. 31, no. 6, pp. 534-535, 2004.

[5] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[6] S. M. LaValle, "Sampling-based motion planning," in *Planning Algorithms*, Cambridge University Press, Cambridge, UK, 2006, http://planning.cs.uiuc.edu/.

[7] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[8] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Technical Report, Computer Science Dept., Iowa State University, 1998.

[9] J. J. Kuffner Jr. and S. M. la Valle, "RRT-connect: an efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 995–1001, April 2000.

[10] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[11] J.-P. Laumond, P. E. Jacobs, M. Taïx, and R. M. Murray, "A Motion Planner for Nonholonomic Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 577–593, 1994.

[12] J. P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot motion planning and control*, vol. 229 of *Lect. Notes Control Inf. Sci.*, pp. 1–53, Springer, London, UK, 1998.

[13] G. Lafferriere and H. Sussmann, "Motion planning for controllable systems without drift," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pp. 1148–1153, April 1991.

[14] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: steering using sinusoids," *Institute of Electrical and Electronics Engineers. Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

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

[16] D. Tilbury, R. M. Murray, and S. S. Sastry, "Trajectory generation for the *n*-trailer problem using Goursat normal form," *Institute of Electrical and Electronics Engineers. Transactions on Automatic Control*, vol. 40, no. 5, pp. 802–819, 1995.

[17] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[18] P. Rouchon, M. Fliess, J. Lévine, and P. Martin, "Flatness, motion planning and trailer systems," in *Proceedings of the 32nd IEEE Conference on Decision and Control*, pp. 2700–2705, IEEE, San Antonio, Tex, USA, December 1993.

[19] S. Sekhavat, F. Lamiraux, J. P. Laumond, G. Bauzil, and A. Ferrand, "Motion planning and control for Hilare pulling a trailer: Experimental issues," in *Proceedings of the 1997 IEEE International Conference on Robotics and Automation, ICRA. Part 4 (of 4)*, pp. 3306–3311, April 1997.

[20] F. Lamiraux and J.-P. Laumond, "Flatness and small-time controllability of multibody mobile robots: application to motion planning," *Institute of Electrical and Electronics Engineers. Transactions on Automatic Control*, vol. 45, no. 10, pp. 1878–1881, 2000.

[21] F. Lamiraux and J.-P. Laumond, "Smooth motion planning for car-like vehicles," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 498–502, 2001.

[22] C. P. Tang, P. T. Miller, V. N. Krovi, J. Ryu, and S. K. Agrawal, "Kinematic control of nonholonomic wheeled mobile manipulator: a differential flatness approach," in *Proceedings of the ASME 2008 Dynamic Systems and Control Conference*, pp. 1117–1124, Ann Arbor, Mich, USA, 2008.

[23] C. P. Tang, "Differential flatness-based kinematic and dynamic control of a differentially driven wheeled mobile robot," in *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics, ROBIO 2009*, pp. 2267–2272, December 2009.

[24] C. P. Tang, P. T. Miller, V. N. Krovi, J.-C. Ryu, and S. K. Agrawal, "Differential-flatness-based planning and control of a wheeled mobile manipulator-theory and experiment," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 4, pp. 768–773, 2011.

[25] C. Fernandes, L. Gurvits, and Z. X. Li, "A variational approach to optimal nonholonomic motion planning," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pp. 680–685, April 1991.

[26] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *The American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

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

[28] H. J. Sussmann and G. Tang, "Shortest paths for the reeds-shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control," Technical Report SYCON-91-10, Department of Mathematics, Rutgers University, September 1991.

[29] P. Souères and J.-P. Laumond, "Shortest paths synthesis for a car-like robot," *Institute of Electrical and Electronics Engineers. Transactions on Automatic Control*, vol. 41, no. 5, pp. 672–688, 1996.

[30] P. Robuffo Giordano, M. Vendittelli, J.-P. Laumond, and P. Souères, "Nonholonomic distance to polygonal obstacles for a car-like robot of polygonal shape," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 1040–1047, 2006.

[31] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.

[32] D. J. Walton and D. S. Meek, "A controlled clothoid spline," *Computers and Graphics (Pergamon)*, vol. 29, no. 3, pp. 353–363, 2005.

[33] A. Piazzi, C. G. Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic G2-splines for the iterative steering of vision-based autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 27–36, 2002.

[34] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouvelière, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 589–606, 2010.

[35] J.-W. Lee and B. Litkouhi, "A unified framework of the automated lane centering/changing control for motion smoothness adaptation," in *Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems, ITSC 2012*, pp. 282–287, September 2012.

[36] L. Z. Wang, K. T. Miura, E. Nakamae, T. Yamamoto, and T. J. Wang, "An approximation approach of the clothoid curve defined in the interval [0, $\pi/2$] and its offset by free-form curves," *CAD Computer Aided Design*, vol. 33, no. 14, pp. 1049–1058, 2001.

[37] J. Sánchez-Reyes and J. M. Chacón, "Polynomial approximation to clothoids via s-power series," *CAD Computer Aided Design*, vol. 35, no. 14, pp. 1305–1313, 2003.

[38] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Söderkvist, "Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 167–172, 2010.

[39] K. Lee, D. Kim, W. Chung, H. W. Chang, and P. Yoon, "Car parking control using a trajectory tracking controller," in *Proceedings of the 2006 SICE-ICASE International Joint Conference*, pp. 2058–2063, October 2006.

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

[41] B. Müller, J. Deutscher, and S. Grodde, "Continuous curvature trajectory design and feedforward control for parking a car," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 541–553, 2007.

[42] B. Mirtich and J. Canny, "Using skeletons for nonholonomic path planning among obstacles," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pp. 2533–2540, May 1992.

[43] P. Zips, M. Böck, and A. Kugi, "Optimisation based path planning for car parking in narrow environments," *Robotics and Autonomous Systems*, 2015.

[44] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamic-domain RRTs: efficient exploration by controlling the sampling domain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3856–3861, Barcelona, Spain, April 2005.

[45] D. Kiss and D. Papp, "Effective navigation in narrow areas: A planning method for autonomous cars," in *Proceedings of the 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 000423–000430, Herl'any, Slovakia, January 2017.

[46] D. Kiss and G. Tevesz, "A steering method for the kinematic car using C*CS paths," in *Proceedings of the 15th International Carpathian Control Conference, ICCC 2014*, pp. 227–232, May 2014.

[47] Á. Nagy, G. Csorvási, and D. Kiss, "Path planning and control of differential and car-like robots in narrow environments," in

Proceedings of the 13th IEEE International Symposium on Applied Machine Intelligence and Informatics, SAMI 2015, pp. 103–108, January 2015.

[48] G. Csorvási, Á. Nagy, and D. Kiss, "RTR+C∗CS: An effective geometric planner for car-like robots," in *Proceedings of the 2015 16th IEEE International Carpathian Control Conference, ICCC 2015*, pp. 85–90, May 2015.

[49] D. Kiss, G. Csorvási, and Á. Nagy, "A planning method to obtain good quality paths for autonomous cars," in *Proceedings of the 4th Eastern European Regional Conference on the Engineering of Computer-Based Systems, ECBS-EERC 2015*, pp. 104–110, August 2015.

[50] D. K. Wilde, "Computing clothoid segments for trajectory generation," in *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 2440–2445, October 2009.

[51] J.-D. Boissonmat, A. Cerezo, and J. Leblond, "A note on shortest paths in the plane subject to a constraint on the derivative of the curvature," Technical Report RR-2160, Inst. Nat. de Recherche en Informatique en Automatique, 1994.

[52] S. Pandi, F. H. P. Fitzek, C. Lehmann et al., "Joint design of communication and control for connected cars in 5G communication systems," in *Proceedings of the 2016 IEEE Globecom Workshops, GC Wkshps 2016*, December 2016.

[53] H. Charaf, G. Harsányi, A. Poppe et al., "BME VIK annual research report on electrical engineering and computer science 2016," *Periodica Polytechnica Electrical Engineering and Computer Science*, vol. 61, no. 2, pp. 83–115, 2017.